

# Queste de savoir

Les réseaux de zéro

---

lundi 02 juin 2025



# Table des matières

Introduction	14
I. Le concept et les bases	17
Introduction	18
I.1. Les réseaux: présentation générale	19
Introduction	19
I.1.1. C'est quoi un réseau ? Ça sert à quoi ?	19
I.1.2. Internet et réseaux privés	20
I.1.2.1. Internet	21
I.1.2.2. Réseaux privés	21
Conclusion	22
I.2. Construire un réseau: le matériel	23
Introduction	23
I.2.1. Les moyens utilisés (médias d'accès)	23
I.2.1.1. Les câbles	23
I.2.1.2. Le monde du sans-fil	25
I.2.2. Le plus important de tous : la carte réseau	27
I.2.3. Concentrateur (hub)	28
I.2.4. Commutateur (switch) et routeur : si peu ressemblants et si similaires	29
I.2.4.1. Le commutateur : juste une histoire d'échange de données	29
I.2.4.2. Le routeur, un véritable ordinateur	29
I.2.5. Répéteur	30
I.2.6. Bilan des matériels	30
Conclusion	31
I.3. Les topologies	32
Introduction	32
I.3.1. Avant tout...	32
I.3.1.1. LAN : le réseau local	32
I.3.1.2. WAN : le réseau étendu	33
I.3.2. C'est quoi une topologie ?	33
I.3.2.1. Topologie physique	34
I.3.2.2. Topologie logique	34
I.3.3. Réseau en bus	34
I.3.4. Topologie de type étoile	35
I.3.5. Réseau en anneau : le ring, mais pas de boxe	36

I.3.6.	Topologie maillée . . . . .	37
I.3.7.	Topologie hybride . . . . .	38
	Conclusion . . . . .	38
	Conclusion	40
II.	Un modèle qui en tient une couche	41
	Introduction	42
II.1.	Introduction aux protocoles	43
	Introduction . . . . .	43
II.1.1.	Vous avez dit protocole ? . . . . .	43
II.1.1.1.	Le protocole : un genre de langue . . . . .	43
II.1.2.	L'utilité d'un protocole par l'exemple . . . . .	44
II.1.3.	Les exigences d'un protocole . . . . .	46
	Conclusion . . . . .	48
II.2.	Ils en tiennent une couche : OSI et TCP/IP	49
	Introduction . . . . .	49
II.2.1.	Le modèle OSI en douceur . . . . .	49
II.2.1.1.	Qu'est-ce que le modèle OSI ? . . . . .	49
II.2.2.	Le modèle OSI par l'exemple : le facteur . . . . .	51
II.2.3.	Survol des couches du modèle OSI . . . . .	52
II.2.3.1.	Comment ça fonctionne ? . . . . .	52
II.2.3.2.	Résumé . . . . .	55
II.2.3.3.	Processus de transmission/réception . . . . .	55
II.2.4.	TCP/IP vs OSI : le verdict ? . . . . .	56
II.2.4.1.	Il y a une génération... . . . . .	56
II.2.4.2.	Comparaison dans la structure . . . . .	57
II.2.4.3.	Point vocabulaire : les unités de données . . . . .	60
II.2.4.4.	Faites attention à l'abstraction des noms de couches . . . . .	61
II.2.4.5.	Critiques du modèle OSI . . . . .	61
II.2.4.6.	Critiques du modèle TCP/IP . . . . .	62
II.2.4.7.	Et maintenant : le verdict des juges . . . . .	62
II.2.5.	Principe d'encapsulation . . . . .	63
	Conclusion . . . . .	67
II.3.	De l'application à la session	68
	Introduction . . . . .	68
II.3.1.	Rôle des couches . . . . .	68
II.3.1.1.	Couche 7 : application . . . . .	68
II.3.1.2.	Couche 6 : présentation . . . . .	69
II.3.1.3.	Couche 5 : le gestionnaire de session . . . . .	70
II.3.2.	BitTorrent, le protocole de partage . . . . .	72
II.3.2.1.	La naissance de BitTorrent . . . . .	72
II.3.2.2.	Le fonctionnement de BitTorrent . . . . .	72
II.3.2.3.	La terminologie de BitTorrent . . . . .	73

II.3.3.	SMTP : le protocole de transmission de mail . . . . .	75
II.3.3.1.	Présentation rapide de SMTP . . . . .	75
II.3.3.2.	Cheminement d'un courriel . . . . .	76
II.3.3.3.	Commençons par le MUA . . . . .	78
II.3.3.4.	C'est quoi, ce MSA ? . . . . .	78
II.3.3.5.	MTA, l'agent de Jack-Bauer de transfert . . . . .	79
II.3.3.6.	Pour terminer : le grand MDA . . . . .	80
II.3.3.7.	Quand les protocoles s'emmêlent... . . . . .	81
II.3.4.	IMAP vs POP : les protocoles de retrait de mail . . . . .	81
II.3.4.1.	Le bureau de poste version électronique : présentation . . . . .	82
II.3.4.2.	IMAP : un protocole qui a la tête dans les nuages . . . . .	82
	Conclusion . . . . .	83
	Conclusion . . . . .	84
III.	Veuillez vous identifier pour communiquer . . . . .	85
	Introduction . . . . .	86
III.1.	Des adresses en folie ! . . . . .	87
	Introduction . . . . .	87
III.1.1.	IP vs MAC . . . . .	87
III.1.1.1.	Adresse IP : l'adresse relative au réseau . . . . .	87
III.1.1.2.	Adresses MAC : l'adresse relative à la carte réseau . . . . .	88
III.1.1.3.	En résumé... . . . .	89
III.1.2.	Masque de sous-réseau et passerelle . . . . .	89
III.1.2.1.	Les sous-réseaux et leurs masques . . . . .	90
III.1.2.2.	...La passerelle . . . . .	91
III.1.3.	Le client et le serveur . . . . .	92
	Conclusion . . . . .	92
	Contenu masqué . . . . .	93
III.2.	Les masques de sous-réseaux : à la découverte du subnetting . . . . .	94
	Introduction . . . . .	94
III.2.1.	En bref . . . . .	94
III.2.2.	L'importance des masques . . . . .	94
III.2.2.1.	Relation entre network ID et masques . . . . .	94
III.2.2.2.	Des règles fondamentales à connaître absolument . . . . .	95
III.2.3.	Introduction au subnetting . . . . .	95
III.2.3.1.	Délégation de l'administration . . . . .	96
III.2.3.2.	La réduction du trafic . . . . .	96
III.2.3.3.	La facilité du diagnostic . . . . .	96
III.2.3.4.	L'économie d'adresses . . . . .	96
III.2.4.	Analyse des contraintes et plan d'adressage . . . . .	97
III.2.4.1.	Analyse des contraintes . . . . .	97
III.2.4.2.	Le prix . . . . .	97
III.2.4.3.	L'évolution du réseau . . . . .	97
III.2.4.4.	Le nombre d'adresses IP . . . . .	98

III.2.4.5. L'organisation . . . . .	98
Conclusion . . . . .	100
III.3. Le subnetting en pratique . . . . .	101
Introduction . . . . .	101
III.3.1. Comment ? . . . . .	101
III.3.1.1. Le comment du pourquoi . . . . .	101
III.3.2. À partir du nombre de sous-réseaux désirés . . . . .	102
III.3.2.1. Exemple de subnetting . . . . .	103
III.3.3. À partir du nombre d'adresses d'hôtes désirées . . . . .	105
III.3.3.1. Explications sur l'adresse de <i>broadcast</i> et l'identité du réseau . . . . .	105
III.3.3.2. Un autre exemple de subnetting . . . . .	106
III.3.3.3. Exemple de subnetting avec moins de 254 hôtes par sous-réseau . . . . .	107
III.3.4. La notation du masque . . . . .	107
III.3.4.1. La notation « classique » . . . . .	108
III.3.4.2. La notation avec un slash ( / ) . . . . .	108
Conclusion . . . . .	108
Contenu masqué . . . . .	108
III.4. La passerelle : les bases du routage . . . . .	110
Introduction . . . . .	110
III.4.1. Une petite révision . . . . .	110
III.4.2. Mode de fonctionnement . . . . .	111
III.4.2.1. Le fonctionnement . . . . .	111
III.4.3. ANDing (conjonction logique) . . . . .	112
III.4.3.1. Le ANDing, approche théorique . . . . .	112
III.4.3.2. Le ANDing par l'exemple . . . . .	113
Conclusion . . . . .	115
III.5. L'adressage par classes (obsolète) . . . . .	116
Introduction . . . . .	116
III.5.1. C'est quoi une classe ? . . . . .	116
III.5.2. Classe A . . . . .	118
III.5.2.1. Présentation . . . . .	118
III.5.2.2. Structure d'une adresse IP de la classe A . . . . .	120
III.5.3. Classes B et C . . . . .	121
III.5.3.1. Classe B . . . . .	121
III.5.3.2. Classe C . . . . .	122
III.5.4. Classes D et E . . . . .	123
III.5.4.1. Quelques informations... . . . .	123
III.5.5. Notion de classe privée . . . . .	124
Conclusion . . . . .	124
Contenu masqué . . . . .	125
III.6. L'adressage CIDR . . . . .	126
Introduction . . . . .	126
III.6.1. Révision de l'adressage par classes . . . . .	126
III.6.2. CIDR et le supernetting . . . . .	127
III.6.2.1. CIDR : le comment . . . . .	128

III.6.2.2.	Comment résumer une route . . . . .	129
III.6.2.3.	Quelques exercices pour la route . . . . .	130
III.6.3.	Les masques à longueurs variables (VLSM) . . . . .	132
III.6.3.1.	Son utilité ? . . . . .	132
III.6.3.2.	TD : implémentation des masques à longueurs variables . . . . .	134
III.6.3.3.	Conclusion et exercices . . . . .	140
III.6.3.4.	Ci-dessous les images d'illustration . . . . .	141
Conclusion	. . . . .	143
Contenu masqué	. . . . .	144
III.7.	Et IPv6 dans tout ça ? . . . . .	146
Introduction	. . . . .	146
III.7.1.	Un format particulier . . . . .	146
III.7.1.1.	Lire une adresse IPv6 . . . . .	147
III.7.2.	L'adresse privée de sortie . . . . .	148
III.7.3.	Les adresses qui voyagent . . . . .	150
III.7.4.	Les adresses particulières . . . . .	151
III.7.4.1.	Les adresses courtes . . . . .	151
III.7.4.2.	Les adresses de site local . . . . .	151
III.7.4.3.	Les adresses multicast . . . . .	152
III.7.4.4.	Les adresses d'encapsulation 4/6 . . . . .	152
Conclusion	. . . . .	152
Conclusion	. . . . .	153
IV.	Dans les basses couches du modèle OSI . . . . .	154
Introduction	. . . . .	155
IV.1.	Introduction à la couche transport . . . . .	156
Introduction	. . . . .	156
IV.1.1.	Présentation . . . . .	156
IV.1.1.1.	L'histoire de Couic le message . . . . .	157
IV.1.1.2.	La relation entre la couche transport et la couche réseau . . . . .	160
IV.1.1.3.	Conclusion . . . . .	166
IV.1.2.	À votre service . . . . .	166
Conclusion	. . . . .	167
IV.2.	Exploration de la couche transport . . . . .	168
Introduction	. . . . .	168
IV.2.1.	Appelle mon numéro... de port . . . . .	168
IV.2.1.1.	Les numéros de port : késaco ? . . . . .	168
IV.2.1.2.	La redirection de port ( <i>port forwarding</i> ) . . . . .	170
IV.2.1.3.	Le scan de port ( <i>port scanning</i> ) . . . . .	170
IV.2.1.4.	Déclenchement de port ( <i>port triggering</i> ) . . . . .	170
IV.2.1.5.	PAT : <i>Port Address Translation</i> . . . . .	171
IV.2.2.	Multiplexing / demultiplexing . . . . .	171
IV.2.2.1.	Structure partielle de l'en-tête de transport . . . . .	172

IV.2.3.	Introduction aux sockets . . . . .	176
IV.2.3.1.	Les fonctions des API . . . . .	178
	Conclusion . . . . .	179
IV.3.	Les protocoles de la couche transport . . . . .	180
	Introduction . . . . .	180
IV.3.1.	UDP, un protocole irresponsable . . . . .	180
IV.3.1.1.	UDP par l'analogie . . . . .	180
IV.3.1.2.	Le principe de la poignée de main . . . . .	182
IV.3.1.3.	La structure d'un datagramme UDP . . . . .	183
IV.3.1.4.	L'essentiel du protocole UDP . . . . .	184
IV.3.2.	Le protocole pessimiste : TCP . . . . .	185
IV.3.2.1.	TCP dans les grandes lignes . . . . .	185
IV.3.2.2.	L'état de connexion avec TCP . . . . .	185
IV.3.2.3.	Étape 1 . . . . .	186
IV.3.2.4.	Étapes 2 et 3 . . . . .	187
IV.3.2.5.	Séquence <del>émotion</del> et acquittement . . . . .	188
IV.3.2.6.	Structure d'un segment . . . . .	189
IV.3.2.7.	Clôture normale . . . . .	190
IV.3.2.8.	Clôture brutale . . . . .	191
IV.3.3.	TCP / UDP : le clash ! . . . . .	192
IV.3.3.1.	Nécessité fonctionnelle . . . . .	192
IV.3.3.2.	Nécessité matérielle . . . . .	193
IV.3.3.3.	Complémentarité . . . . .	194
	Conclusion . . . . .	195
IV.4.	Aujourd'hui, contrôle! . . . . .	196
	Introduction . . . . .	196
IV.4.1.	Le contrôle de flux . . . . .	196
IV.4.1.1.	Acquitté ! . . . . .	197
IV.4.1.2.	Derrière les fenêtres... . . . .	201
IV.4.2.	Le contrôle de congestion . . . . .	203
IV.4.2.1.	Le premier : Tahoe . . . . .	203
IV.4.2.2.	Reno . . . . .	206
IV.4.2.3.	Modification du protocole . . . . .	207
IV.4.3.	Le principe de la somme de contrôle . . . . .	209
IV.4.3.1.	Le principe par l'analogie (encore et toujours) . . . . .	209
IV.4.3.2.	Exemple de computation abstraite . . . . .	210
	Conclusion . . . . .	212
IV.5.	La couche 3: le réseau . . . . .	213
	Introduction . . . . .	213
IV.5.1.	Rôle et matériel . . . . .	213
IV.5.1.1.	Le routage, qu'est-ce que c'est ? . . . . .	213
IV.5.1.2.	Rôle de la couche . . . . .	213
IV.5.1.3.	Et le matos ? . . . . .	214
IV.5.2.	Les codes de la route . . . . .	215
IV.5.2.1.	La table de routage . . . . .	215
IV.5.2.2.	N'y allons pas par quatre chemins . . . . .	215

IV.5.2.3.	À l'aide, j'ai perdu ma route ! . . . . .	217
IV.5.2.4.	Afficher le contenu de la table . . . . .	217
IV.5.2.5.	Les différents types de routes . . . . .	218
IV.5.3.	Quand utilise-t-on la passerelle ? . . . . .	220
IV.5.3.1.	La procédure de routage . . . . .	222
IV.5.4.	C'est quoi, ton type de routage ? . . . . .	224
IV.5.4.1.	Unicast : un seul destinataire . . . . .	224
IV.5.4.2.	Multicast : restriction à un groupe . . . . .	225
IV.5.4.3.	Le broadcast . . . . .	225
IV.5.4.4.	Anycast : à n'importe qui ? . . . . .	225
IV.5.5.	Introduction aux protocoles de routage . . . . .	228
IV.5.5.1.	Routage statique . . . . .	228
IV.5.5.2.	Routage dynamique . . . . .	228
	Conclusion . . . . .	229
IV.6.	Le routage par l'exemple . . . . .	231
	Introduction . . . . .	231
IV.6.1.	En route, mauvaises troupes ! . . . . .	231
IV.6.2.	Dans le grand bain d'Internet . . . . .	233
	Conclusion . . . . .	234
IV.7.	Les protocoles de routage . . . . .	235
	Introduction . . . . .	235
IV.7.1.	Direct au cimetière avec RIP . . . . .	235
IV.7.1.1.	Première version . . . . .	235
IV.7.1.2.	Alléluia, la v2 est là . . . . .	235
IV.7.1.3.	Côté technique . . . . .	236
IV.7.1.4.	Séparation horizontale . . . . .	236
IV.7.1.5.	Comparaison finale . . . . .	239
IV.7.2.	Vecteur de distances . . . . .	239
IV.7.2.1.	Le principe d'un algorithme VD . . . . .	240
IV.7.2.2.	Déterminons les chemins actuels . . . . .	241
IV.7.2.3.	Premier échange des VD . . . . .	244
IV.7.2.4.	Deuxième échange des VD . . . . .	247
IV.7.2.5.	Suite et fin de l'algorithme . . . . .	249
IV.7.3.	OSPF, le mastodonte . . . . .	250
IV.7.4.	Protocoles à états de lien . . . . .	252
IV.7.4.1.	Notions de graphe . . . . .	252
IV.7.4.2.	Qui ça ? . . . . .	256
IV.7.5.	BGP : les entrailles d'Internet . . . . .	260
	Conclusion . . . . .	261
IV.8.	Le protocole IP . . . . .	262
	Introduction . . . . .	262
IV.8.1.	L'en-tête IPv4 . . . . .	262
IV.8.2.	L'en-tête IPv6 . . . . .	266
IV.8.3.	La fragmentation . . . . .	269
IV.8.4.	ICMP, l'ange gardien . . . . .	271
	Conclusion . . . . .	274

Conclusion	275
<b>V. On touche le fond !</b>	<b>276</b>
Introduction	277
<b>V.1. Les liaisons dangereuses</b>	<b>278</b>
Introduction	278
V.1.1. Données, données, do-o-nnées...	278
V.1.2. Les interfaces	279
Conclusion	283
<b>V.2. Qu'est-ce qui se trame chez Ethernet ?</b>	<b>284</b>
Introduction	284
V.2.1. Un réseau Ethernet, kékako ?	284
V.2.2. Le protocole	285
V.2.3. Et VLAN !	287
V.2.4. 1, 2, 3, on switch !	291
V.2.4.1. Réception	291
V.2.4.2. Commutation	291
V.2.4.3. Transmission	292
Conclusion	292
<b>V.3. Deux points c'est tout</b>	<b>293</b>
Introduction	293
V.3.1. Tout vient à point...	293
V.3.2. ... à qui sait encapsuler	294
V.3.3. Le bout du tunnel	296
V.3.3.1. Au sec dans le tunnel	297
V.3.3.2. Vers Internet et au-delà	298
Conclusion	299
<b>V.4. MPLS, un protocole qui en a le cœur net</b>	<b>300</b>
Introduction	300
V.4.1. Les réseaux d'opérateur privés	300
V.4.2. Mélange des genres	301
Conclusion	304
<b>V.5. Soyez au courant</b>	<b>305</b>
Introduction	305
V.5.1. Attention à la trame !	305
V.5.1.1. La transmission dans les grandes lignes	305
V.5.1.2. Quels signaux ?	306
V.5.1.3. Et avec d'autres supports ?	306
V.5.2. Fais le baud	306
V.5.2.1. Le plus simple : NRZ	307
V.5.2.2. MLT-3	307
V.5.2.3. Manchester	308

V.5.2.4. Et maintenant, une astuce . . . . .	309
Conclusion . . . . .	309
V.6. Remontons à la surface . . . . .	310
Introduction . . . . .	310
V.6.1. Contexte . . . . .	310
V.6.2. Au niveau de la carte réseau . . . . .	310
V.6.3. Au niveau du processus . . . . .	312
V.6.4. Au niveau de l'application . . . . .	314
Conclusion . . . . .	315
Conclusion . . . . .	316
VI. Reprenons du service . . . . .	317
Introduction . . . . .	318
VI.1. DNS arrive à point nommé . . . . .	319
Introduction . . . . .	319
VI.1.1. En bref . . . . .	319
VI.1.2. Un arbre à l'envers . . . . .	320
VI.1.2.1. Passe à ton voisin . . . . .	321
VI.1.2.2. Par la racine . . . . .	322
VI.1.3. Drôle de type . . . . .	325
VI.1.3.1. Association d'adresses IP . . . . .	326
VI.1.3.2. Services particuliers . . . . .	326
VI.1.3.3. Allez voir ailleurs . . . . .	326
VI.1.3.4. Divers . . . . .	326
VI.1.3.5. Récapitulatif . . . . .	327
VI.1.4. Évolutions . . . . .	327
VI.1.4.1. DNSSEC . . . . .	327
VI.1.4.2. DOT . . . . .	328
VI.1.4.3. DOH . . . . .	328
Conclusion . . . . .	328
VI.2. DHCP, un service rempli d'adresses . . . . .	330
Introduction . . . . .	330
VI.2.1. Côté client . . . . .	330
VI.2.1.1. En IPv4 . . . . .	330
VI.2.1.2. En IPv6 . . . . .	334
VI.2.2. Côté serveur . . . . .	334
VI.2.2.1. Fonctionnement des serveurs . . . . .	334
VI.2.2.2. Agent à votre service . . . . .	336
Conclusion . . . . .	336
VI.3. NAT ou l'art de la dissimulation . . . . .	337
Introduction . . . . .	337
VI.3.1. NAT : cette adresse que je ne saurais voir . . . . .	337

VI.3.2.	PAT : lost in translation . . . . .	338
VI.3.3.	Le port forwarding . . . . .	340
	Conclusion . . . . .	341
VI.4.	Le proxy dans tous ses états . . . . .	342
	Introduction . . . . .	342
VI.4.1.	Principe . . . . .	342
VI.4.2.	Aspect réseau . . . . .	343
VI.4.3.	Limites et contournement . . . . .	345
	Conclusion . . . . .	351
VI.5.	Dans le collimateur de la supervision . . . . .	352
	Introduction . . . . .	352
VI.5.1.	Intérêt en cas de dommage . . . . .	353
VI.5.2.	Comment ça marche ? . . . . .	353
VI.5.3.	Exemples de superviseurs . . . . .	357
	VI.5.3.1. Zabbix . . . . .	357
	VI.5.3.2. DX Spectrum . . . . .	357
	VI.5.3.3. Nagios . . . . .	358
	Conclusion . . . . .	359
	Conclusion . . . . .	360
VII.	Évolutions . . . . .	361
	Introduction . . . . .	362
VII.1.	Saint Cloud, priez pour nous . . . . .	363
	Introduction . . . . .	363
VII.1.1.	Une pluie de services . . . . .	363
	VII.1.1.1. Infrastructure . . . . .	364
	VII.1.1.2. Plateforme et fonction . . . . .	364
	VII.1.1.3. Logiciel . . . . .	365
	VII.1.1.4. Stratégie . . . . .	366
VII.1.2.	Modèle économique . . . . .	367
	VII.1.2.1. La mutualisation . . . . .	367
	VII.1.2.2. Les datacentres . . . . .	369
	VII.1.2.3. Sur le bord . . . . .	370
	Conclusion . . . . .	372
VII.2.	Software Defined Networking: le réseau à la carte . . . . .	373
	Introduction . . . . .	373
VII.2.1.	Un réseau plan-plan . . . . .	373
	VII.2.1.1. Gestionnaire de trames . . . . .	375
	VII.2.1.2. Le protocole OpenFlow . . . . .	378
VII.2.2.	Tout un programme . . . . .	378
	Conclusion . . . . .	378

Conclusion	380
VIII. Quelques notions de sécurité réseau	381
Introduction	382
VIII.1. Introduction à la sécurité	383
Introduction	383
VIII.1.1. C'est quoi la sécurité ?	383
VIII.1.1.1. Étude de cas : Zeste de Savoir	384
VIII.1.2. Comprendre la terminologie	385
VIII.1.2.1. Une menace	385
VIII.1.2.2. La vulnérabilité	385
VIII.1.2.3. Les étapes d'exploitation d'une faille	386
VIII.1.2.4. Cible d'évaluation	387
VIII.1.2.5. Qu'est-ce qu'une attaque ?	387
VIII.1.2.6. Identifier / Authentifier / Autoriser	388
VIII.1.3. Les moyens de sécurité	389
VIII.1.3.1. La sécurité physique	389
VIII.1.3.2. Les techniques de contrôle d'accès	389
Conclusion	390
VIII.2. Malins, les logiciels !	391
Introduction	391
VIII.2.1. Des fins malveillantes	391
VIII.2.1.1. Les adwares : du spam directement sur votre machine !	391
VIII.2.1.2. Les keyloggers : dis-moi ce que tu tapes, je te dirai que tu es ma victime	391
VIII.2.1.3. Les backdoors : c'est donc ça, ce courant d'air...	391
VIII.2.1.4. Les espions : la curiosité est un vilain défaut, mais qui peut rapporter gros	392
VIII.2.1.5. Les trojans : ils s'invitent tous seuls, c'est trojantil !	392
VIII.2.2. Ils n'ont pas volé leurs répliques : les virus et les vers	392
VIII.2.2.1. Le ver : il se duplique pour ne pas être solitaire	392
VIII.2.2.2. Le virus : il se cache pour ne pas se faire virer	392
VIII.2.3. Ils s'incrument au cœur du système !	393
Conclusion	393
VIII.3. L'attaque de l'homme du milieu (MITM)	394
Introduction	394
VIII.3.1. Le protocole ARP ?	394
VIII.3.2. La faiblesse de ce protocole	395
VIII.3.2.1. Le MITM : quand un intrus s'en mêle...	395
VIII.3.3. Exemple concret	398
VIII.3.4. Peut-on se défendre ?	401
Conclusion	402
VIII.4. Allumez le pare-feu	403
Introduction	403

VIII.4.1. C'est quoi ?	403
VIII.4.1.1. Pare-feu logiciel	403
VIII.4.1.2. Pare-feu matériel	404
VIII.4.2. Sans filtre	405
VIII.4.3. Autres fonctionnalités	407
Conclusion	407
Conclusion	409
<b>IX. Annexes</b>	<b>410</b>
Introduction	411
<b>IX.1. Binaire et hexadécimal : partez sur de bonnes bases !</b>	<b>412</b>
Introduction	412
IX.1.1. Décimal vs binaire : un peu de pratique	412
IX.1.1.1. Système décimal	412
IX.1.1.2. Système binaire	413
IX.1.1.3. C'est parti, touchons du binaire	413
IX.1.2. Un point sur l'hexadécimal	416
Conclusion	418
Contenu masqué	418
<b>IX.2. La détection d'erreurs avec la somme de contrôle</b>	<b>419</b>
Introduction	419
IX.2.1. La vérification de parité	419
IX.2.1.1. Le bit de parité pair	420
IX.2.1.2. Le bit de parité impair	420
IX.2.1.3. Exercices sur le bit de parité pair	421
IX.2.1.4. Exercices sur le bit de parité impair	421
IX.2.1.5. Conclusion	422
IX.2.2. La somme de contrôle de Fletcher	422
IX.2.2.1. Un Fletcher sauvage apparaît !	423
IX.2.3. L'algorithme Adler-32	427
IX.2.4. Quand IP rime avec simplicité	429
Conclusion	431
Contenu masqué	431
<b>IX.3. Analyse fine des communications réseaux</b>	<b>432</b>
Introduction	432
IX.3.1. Présentation générale	432
IX.3.2. Utilisation de tcpdump	432
IX.3.3. Utilisation de Wireshark	435
Conclusion	436
<b>IX.4. Qui gouverne Internet ?</b>	<b>438</b>
Introduction	438
IX.4.1. Des chiffres et des lettres	438

## *Table des matières*

IX.4.2.	Des standards et des normes . . . . .	440
IX.4.3.	Des politiques et des lois . . . . .	443
	Conclusion . . . . .	443
	<b>Conclusion</b>	<b>444</b>

# Introduction

Salut les agrumes ! 🍊

Vous êtes curieux ou passionné par les réseaux informatiques ? Vous êtes étudiant et avez du mal à appréhender et comprendre cet univers ? Vous êtes tombé au bon endroit ! Nous vous souhaitons la bienvenue dans ce tutoriel. Entrons tout de suite dans le vif du sujet. Avez-vous déjà entendu une discussion d'administrateurs réseau ? Vous avez probablement entendu des termes compliqués, qui vous semblent insignifiants comme réseau privé virtuel, protocole, niveau applicatif, UDP, transmission de paquets... Et alors quand ils parlent de leurs problèmes de configuration de routeurs, de passerelles, de serveurs DNS, vous vous dites qu'ils viennent d'une autre planète ! Rassurez-vous, ce tutoriel est là pour vous expliquer comment tout cela fonctionne.

Pour commencer, vous appréhendez les bases du réseau informatique. Puis vous découvrirez la notion fondamentale de protocole. Ensuite, vous verrez comment on s'y repère sur un réseau avec l'adressage. Une fois ces bonnes bases acquises, vous pourrez vous focaliser sur les mécanismes avancés de la communication. L'aspect physique des transmissions vous sera aussi présenté. Finalement, vous vous rendrez compte avec les services à quel point tout est fait pour que l'utilisation des réseaux soit facilitée. 🍊

En fin de tutoriel, vous trouverez quelques notions de sécurité réseau ainsi que des sujets annexes. Ces derniers sont des concepts moins techniques, voire assez généraux mais indispensables en réseau. Ces chapitres peuvent être feuilletés indépendamment du reste du cours.



### Licence et réutilisation

Depuis fin 2011, ce cours est sous licence [Creative Commons BY-NC-ND 2.0](#) 📄 . Vous pouvez copier tout ou une partie de ce tutoriel pour un cours d'université, un exposé, un TPE, etc. à condition de citer au moins l'adresse de ce cours. L'utilisation commerciale et la modification sont interdites. Des autorisations exceptionnelles peuvent être demandées à l'adresse [reseauxdezero \(at\) gmail \(point\) com](mailto:reseauxdezero(at)gmail(point)com) en précisant ce que vous souhaitez faire de ce cours. Nous vous répondrons le plus rapidement possible.

Les images présentées sans crédit sont soumises à la même licence que le cours, excepté l'icône du tuto (CC BY). Sauf mention contraire, les images sous licence [CC BY](#) 📄 sont de Titouan Soulard, également co-auteur de [la Cybersécurité de Zéro](#) 📄 .

Pour toute question, vous pouvez envoyer un e-mail à l'adresse ci-dessus, nous contacter sur [notre page Facebook](#) 📄 et [notre compte Twitter](#) 📄 . Les membres de Zeste de Savoir peuvent également commenter ce tutoriel ou envoyer un message privé @Vince.



### Allez encore plus loin !

Vous préférez lire sur format papier ? **Les Réseaux de Zéro** sont disponibles en livre aux éditions Eyrolles ! 📄 Cette édition comporte en plus des exercices pratiques sur simulateur, pour configurer du matériel professionnel et vous arracher les cheveux sur des problèmes dont vous vous souviendrez longtemps ! 🍊 Vous êtes plutôt sécurité réseau et hacking ? **La Cybersécurité de Zéro** 📄 est le livre qu'il vous faut !



FIGURE 1. – La Cybersécurité de Zéro et Les Réseaux de Zéro ne demandent qu'à rejoindre votre bibliothèque !

# Première partie

## Le concept et les bases

# Introduction

Bien, après une introduction prometteuse, nous allons enfin commencer. Dans cette partie nous allons apprendre beaucoup de théorie. Nous allons dans un premier temps nous attarder sur cette question : qu'est-ce qu'un réseau ? Nous étudierons et comprendrons ce que c'est qu'un réseau. Nous allons voir que les réseaux n'existent pas qu'en informatique. Nous verrons de quoi est composé un réseau, le matériel nécessaire, et la forme qu'un réseau peut prendre. On commence donc doucement, mais sûrement !

# I.1. Les réseaux : présentation générale

## Introduction

Dans ce chapitre, nous allons aborder la notion de réseau, en commençant par une question toute simple : c'est quoi un réseau ?

### I.1.1. C'est quoi un réseau ? Ça sert à quoi ?

Avant toute chose, il est indispensable de répondre à la question suivante : qu'est-ce qu'un réseau ? On pourrait définir le mot « réseau » en une phrase : un réseau est un groupe d'entités en communication.



C'est quoi une entité ?

Une entité peut désigner une « chose » parmi d'autres. Par exemple, une personne dans un groupe de personnes est une entité de ce groupe. Pour rester dans cet exemple, on parle de réseau quand deux ou plusieurs personnes parlent ensemble.



C'est tout, un réseau c'est juste quand on parle ensemble ?

Oui, mais n'oubliez pas que « parlent ensemble » c'est aussi « s'échangent des informations » ! 🍊 Donc, en gros, un réseau consiste en l'échange d'informations, et il existe (dans la vie courante) plusieurs moyens d'échanges d'informations, sans faire intervenir la technologie (Internet, téléphone, etc.). Si on veut vous donner un livre, on prend le livre, et on vous tend la main, puis vous prenez le livre. 🍊 Vous l'aurez compris, il existe plusieurs manières de partager des données entre les humains, sans les technologies.

Ce qui est intéressant, c'est que je peux envoyer (transmettre) un livre à André, en passant par Pierre.

Eh Pierre, si tu vois André, passe-lui le livre, et qu'il te le remette quand il aura fini de le lire.

Ce qui se passe dans ce cas est :

- Je donne le livre à Pierre
- Pierre trouve André et le lui donne
- André a fini, il rend le livre à Pierre
- Pierre vient me rendre le livre

## I. Le concept et les bases

Nous allons supposer dans ce cas présent qu'André et moi ne nous voyons pas, donc, Pierre est dans ce cas un intermédiaire. Justement, le principe d'intermédiaire est un des fondements des réseaux informatiques. Vous allez rapidement vous en rendre compte.



Pour communiquer, les 2 entités doivent parler la même langue. Ou alors, l'intermédiaire doit parler la langue de chacun de ses interlocuteurs. En réseau informatique, c'est pareil, sauf qu'on ne parle pas de langue mais de protocole.

Si vous avez compris ce qu'est un réseau avec des humains, vous avez tout compris. Un réseau informatique est exactement pareil, sauf qu'il faut remplacer les humains par des machines. Hé oui. 🍊



Mais... mais... et les câbles, les adresses je-ne-sais-pas-quoi... ? On en fait quoi ? 🍊

On ne va pas se compliquer l'existence tout de suite hein. 🍊 Pour l'instant, on reste dans l'approche globale du réseau ; les liaisons et la configuration, on verra plus tard. Vous ne voulez quand même pas que l'on monte un réseau d'entreprise dès le premier chapitre, si ? 🍊



Concrètement, un réseau informatique, ça sert à quoi ?

Eh bien, sans réseau informatique, vous ne seriez pas en train de lire ce tuto, déjà. 🍊 De manière globale, un réseau informatique permet l'échange d'informations à distance. On peut trouver plein d'applications à un réseau : discuter avec une personne, s'échanger des documents, jouer en ligne...



Retenez bien le terme d'application de réseau ! Une application est l'utilisation (voire l'exploitation) d'une ressource pour en faire quelque chose de concret. Ici, on exploite un réseau informatique pour discuter par exemple. En mécanique, on peut exploiter du matériel pour faire une voiture : c'est une application de la mécanique (le rédacteur ayant écrit ça n'y connaît absolument rien en mécanique, si quelqu'un veut refaire l'exemple qu'il n'hésite pas 🍊 ).

### I.1.2. Internet et réseaux privés

Dans un tutoriel sur les réseaux informatiques, on ne pouvait pas manquer de parler d'Internet, bien évidemment. Mais si on le connaît un minimum pour l'utiliser au quotidien, on sait moins qu'une énorme partie des échanges passe par des réseaux parallèles. Ça fait peur, hein ? 🍊

### I.1.2.1. Internet

Internet, c'est toujours compliqué à définir. Alors on va dire ça de manière très pragmatique : Internet, c'est un ensemble de réseaux. Nous voilà bien avancés, hein ? 🍊 C'est pourtant bien une interconnexion de réseaux (*inter-network*) de tous types qui permet de relier le monde. On peut y transmettre toutes sortes de données : vidéo, texte, images, sons, etc. Mais ce qu'il faut garder en tête, c'est que c'est un réseau **public**. Cela veut dire que n'importe qui peut y accéder et communiquer avec n'importe qui. Et cela implique (en théorie) une **neutralité du réseau**. C'est-à-dire que toutes les communications se retrouvent mélangées indistinctement et personne n'est mieux loti qu'un autre.

Vous allez vite vous en rendre compte, dans ce cours, nous faisons beaucoup d'analogies. Faisons un parallèle entre Internet et la voie publique, autrement dit, la rue. Oui, vous savez, le monde dehors qu'on peut voir quand on lève la tête de son téléphone. 🍊 Quand vous vous déplacez dans la rue pour aller de chez vous au supermarché, vous êtes mélangé à tout le monde et vous devez respecter les mêmes règles que tout le monde : s'arrêter au stop pour les véhicules, ne pas traverser n'importe comment quand on est à pied, etc. Pourquoi dit-on cela, alors que ça paraît évident ? Eh bien, parce qu'en dehors d'Internet, c'est pas comme ça. 🍊

### I.1.2.2. Réseaux privés

Quand on parle de réseaux privés, on pense naturellement à des petits réseaux chez soi. Ceux dont on va parler ici n'ont rien à voir. Il faut savoir qu'il existe, chez certains opérateurs de télécommunications, des réseaux distincts qui ne servent qu'aux entreprises, généralement pour relier différents sites entre eux. Quand le PDG d'une grande société est en visioconférence de crise parce qu'un service critique est défaillant dans son entreprise, vaut mieux pas que les employés qui glandent sur les réseaux sociaux ne saturer la connexion ! 🍊 Sur les réseaux privés, on peut librement faire de la discrimination selon les communications. On peut décider par exemple que les visioconférences seront prioritaires en cas d'utilisation importante du réseau.

Pour continuer avec notre analogie précédente, supposons que vous ayez un très grand terrain privé avec des voies de circulation. Eh bien, rien ne vous empêche de décider arbitrairement que sur ce terrain, on circule à gauche et que la vitesse maximale des automobiles est de 200 km/h. C'est chez vous, après tout. Par contre, dès que vous sortez du terrain, vous devrez impérativement respecter les règles de la voie publique. C'est pareil si un réseau privé est interconnecté avec Internet : toute discrimination ou spécificité dans le trafic ne pourra pas se transmettre sur le réseau public.



Il y a bien une distinction physique entre les liens servant à Internet et les liens utilisés pour les réseaux d'entreprise. Mais chaque entreprise ne dispose pas d'infrastructures propres pour interconnecter ses sites ! Les réseaux privés des opérateurs télécoms mutualisent les connexions de leurs clients. Tout ce trafic est bien cloisonné par des procédés logiciels, une entreprise ne risque pas de pouvoir espionner son concurrent. 🍊

## Conclusion

On espère que ce chapitre ne vous a pas ennuyé, car il est primordial si l'on veut avancer dans le cours. Nous avons abordé le fonctionnement de la transmission des données, en nous inspirant de la vie courante. L'exemple n'était certes pas original, mais il est tout de même très pratique pour comprendre les adresses, les protocoles, etc. Vous n'allez pas tarder à vous en rendre compte !

Maintenant que nous avons défini ce qu'est un réseau, nous allons pouvoir étudier de quoi c'est composé. 🍊

## I.2. Construire un réseau : le matériel

### Introduction

Il faut savoir que pour construire un réseau, il faut du matériel. Tout comme il faut un moteur, des roues et autres pour construire une voiture. Nous verrons donc quels sont les appareils et comment ils sont reliés entre eux : câbles, transmission sans fil, etc.

#### I.2.1. Les moyens utilisés (médias d'accès)

En informatique, les médias d'accès sont les moyens utilisés pour rendre possible la communication (l'échange des informations) entre les ordinateurs. Voyons divers moyens de connecter des ordinateurs entre eux.

##### I.2.1.1. Les câbles

Un des médias d'accès les plus utilisés est le câble. Les câbles sont des liaisons physiques entre ordinateurs. Mais il en existe différentes sortes, nous allons en voir 2 principales.

###### I.2.1.1.1. Câble Ethernet

Le câble Ethernet est sûrement le type de câble le plus utilisé pour connecter des ordinateurs entre eux dans un réseau local. À moins que votre réseau soit entièrement sans-fil, vous en avez sûrement chez vous. Il relie généralement un ordinateur personnel à un routeur (ce que l'on appelle parfois une « box »). Le nom formel de ces câbles est **paire torsadée**, en anglais *twisted pair*. À l'intérieur se trouvent en réalité 4 paires de fils de cuivre qui servent notamment aux transmissions électroniques. Il en existe plusieurs catégories, les plus courantes sont la 5, la 5E et la 6. Elles possèdent des caractéristiques physiques différentes qui font varier leur longueur et leur débit. Ainsi, un câble de catégorie 5 ( **CAT5** ) ne peut ni mesurer plus de 100 mètres, ni dépasser les 100 Mb/s. Un câble **CAT5E** peut, pour la même longueur, supporter un débit de 1 Gb/s.



Les paires peuvent être protégées contre les interférences extérieures par une feuille d'aluminium. On parle de **blindage**, en anglais *shield*. On retrouve ce terme dans des acronymes comme UTP- **CAT5** (Unshielded Twisted Pair Category 5).

## I. Le concept et les bases

Il existe deux types de câble Ethernet : les câbles Ethernet droits et les câbles Ethernet croisés. Ces derniers permettent de relier directement entre eux deux ordinateurs alors que les câbles droits servent à relier un ordinateur à un autre appareil comme un *hub* ou un *switch* que nous allons vous présenter dans ce chapitre.

?

Comment faire pour reconnaître un câble droit d'un câble croisé ?

Généralement, c'est marqué sur l'emballage. 🍊

Si vous n'avez plus l'emballage, il suffit de regarder les embouts des câbles :

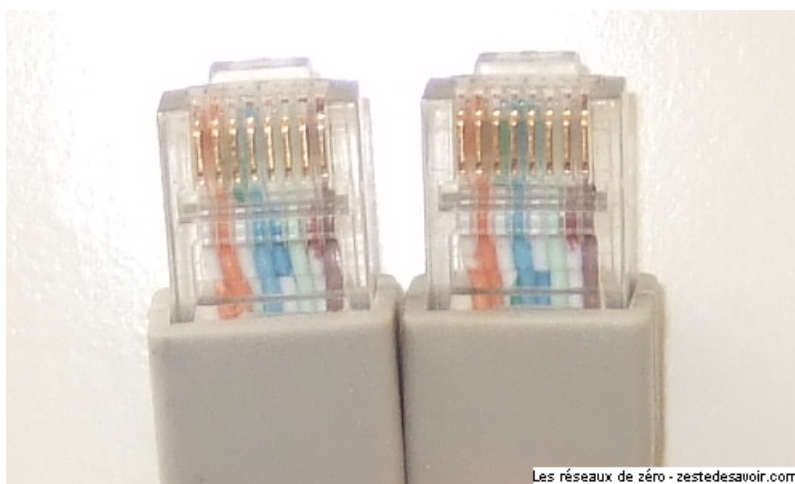


FIGURE I.2.1. – Embouts de câble Ethernet

Sur cette photo, on voit que les couleurs des fils à l'intérieur des embouts sont dans le même ordre sur les deux *connecteurs* : c'est donc un câble **droit**. Si le premier fil en partant de la gauche est inversé avec le 3ème et que le 2ème est inversé avec le 6ème, c'est un câble **croisé**. Sinon, c'est un câble dit « bâtard », mais c'est rare.

Ce type de câble est parfois appelé « câble **RJ45** » : c'est un abus de langage, **RJ45** est le nom de l'*interface* du câble (en gros, son embout).

### I.2.1.1.2. Câble téléphonique

Le câble téléphonique est communément appelé **RJ11**. Ici aussi c'est un abus de langage, **RJ11** n'est pas le câble, mais bien l'interface. C'est ce que l'on peut utiliser pour le téléphone et le modem. En France, ce type de câble est peu utilisé : les prises en T sont très courantes pour les lignes DSL, qui fonctionnent sur des fils de cuivre. Ce type de liaison tend à disparaître au profit de la fibre optique.

Ce tutoriel n'ayant pas pour objectif d'aller dans les détails techniques électroniques qui constituent ces câbles et leurs spécificités, ces notions seront suffisantes pour le moment. Si vous voulez approfondir vos notions sur les câblages, nous vous conseillons Google et Wikipédia. 🍊

### I.2.1.2. Le monde du sans-fil

L'air est aussi un média d'accès en réseau informatique. C'est un espace global qui englobe d'autres médias d'accès, dont nous allons parler. On peut diffuser des ondes électromagnétiques dans l'air et dans l'espace : ce sont ces ondes qui permettent de transporter des informations.

#### I.2.1.2.1. Le Bluetooth

Le Bluetooth, qui signifie littéralement dent bleue (🍌) est une technologie développée par plusieurs entreprises (Agere, IBM, Intel, Microsoft, Motorola, Nokia et Toshiba) permettant la communication en utilisant l'espace hertzien (qui permet la diffusion d'ondes radio) entre les équipements électroniques, afin de minimiser l'utilisation des câbles entre les imprimantes, ordinateurs, scanners, PDA, téléphones, etc. Ce système exploite donc les ondes radio. D'ailleurs, vous allez apprendre du vocabulaire aujourd'hui : quand plusieurs entités sont en communication par le biais du Bluetooth, ce réseau formé est qualifié de **piconet** 🍌. Piconet vient de pico-network, en français on peut traduire ça par picoréseau. Dans un picoréseau, les appareils utilisent la relation **maître-esclave** : le maître donne des ordres, l'esclave obéit. Quand plusieurs picoréseaux sont reliés, les esclaves peuvent avoir plusieurs maîtres, on parle alors de scatternet ou inter-réseau. Le mot « scatternet » signifie littéralement « réseau dispersé ».



En Bluetooth, un esclave ne peut communiquer qu'avec son ou ses maîtres. Il ne peut pas communiquer avec d'autres esclaves ou maîtres. Inversement, un maître ne peut communiquer qu'avec son ou ses esclaves (bien qu'un maître puisse être lui-même esclave d'un autre). D'ailleurs, un maître ne peut pas avoir plus de 7 esclaves.

Voici des schémas expliquant ces 2 types de réseaux (piconet et scatternet) :

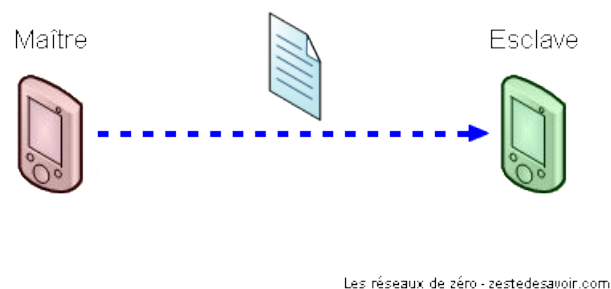


FIGURE I.2.2. – Un piconet, ou picoréseau

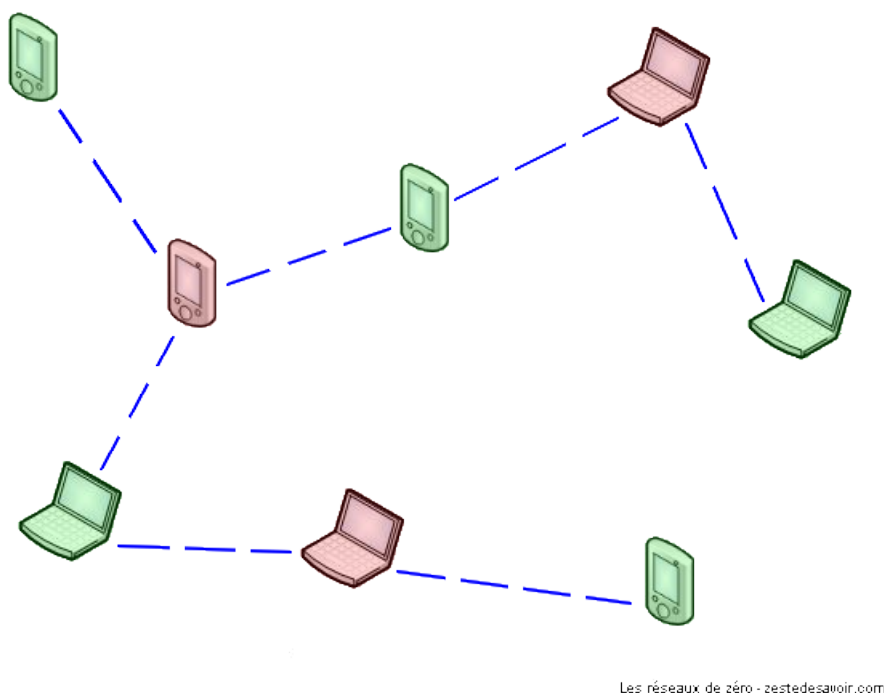


FIGURE I.2.3. – Un scatternet

Pour la petite histoire, le nom Bluetooth vient d'un roi danois qui s'appelait Harald Ier, surnommé Harald Blåtand, ce qui signifie « l'homme à la dent bleue ». 🍊

Il existe 3 classes en Bluetooth : la classe 1, la 2 et la 3. Ce qui les différencie est juste la portée. Dans la classe 1, la portée peut aller jusqu'à 100 mètres, dans la catégorie 2, elle est d'une dizaine de mètres, et dans la classe 3, elle est de quelques mètres seulement (moins de 10). C'est cette 3ème classe qui est utilisée dans les téléphones portables.

#### I.2.1.2.2. L'infrarouge

L'infrarouge est un autre moyen de transmission des données sans fil, qui exploite la lumière. Il est moins pratique que le Bluetooth car il faut que les périphériques qui communiquent entre eux soient à moins de 1,50m de distance. Ils doivent aussi être alignés : la lumière ne se propage pas dans les environs comme les ondes radio. Autrefois, beaucoup de téléphones utilisaient l'infrarouge, mais il s'est rapidement fait remplacer par le Bluetooth, bien que certains appareils utilisent les deux. Il existe toujours actuellement des imprimantes, souris, claviers sans fil utilisant l'infrarouge.

#### I.2.1.2.3. Le Wi-Fi

Le Wi-Fi est certainement le moyen de transmission de données sans fil le plus utilisé. Sa portée pouvant excéder les 200 mètres en espace ouvert et sa vitesse de débit théorique de l'ordre du gigabit par seconde ( Gb/s) font que cette technologie est aujourd'hui très utilisée dans les réseaux locaux pour accéder à Internet. Il est impressionnant de constater le nombre de points d'accès Wi-Fi sécurisés ou non que l'on peut capter un peu partout. « Wi-Fi » peut

être considéré comme le nom commercial de la norme IEEE 802.11, norme qui régit cette technologie.

Ces méthodes de transmission d'information ne serviraient à rien si l'on n'avait pas de matériel pour les exploiter... Heureusement, il y en a, et pas qu'un peu ! 🍊

### 1.2.2. Le plus important de tous : la carte réseau

La carte réseau est le composant le plus important, elle est indispensable. C'est par elle que transitent toutes les données à envoyer et à recevoir du réseau par un ordinateur. Il n'y a pas grand-chose à dire sur cet appareil. La seule chose que vous devez connaître, c'est la notion d'**adresse MAC** : c'est l'adresse physique de la carte. Elle permet d'identifier la machine dans un réseau, un peu comme l'**adresse IP**. Nous ne devrions pas encore en parler, mais il serait bien difficile de comprendre le fonctionnement de certains matériels... Pour faire court et ne pas vous embrouiller si tôt, l'adresse physique est relative à la carte réseau. Elle lui est attribuée à sa fabrication et ne peut pas changer (ce n'est pas tout à fait vrai, mais l'idée est là). L'adresse IP est relative au réseau : elle change tout bonnement suivant le réseau. Vous comprendrez mieux ce que sont ces adresses dans la sous-partie sur le commutateur (*switch*). La carte réseau est aussi appelée NIC en anglais, pour *Network Interface Card*. Voici à quoi peut ressembler une carte réseau :

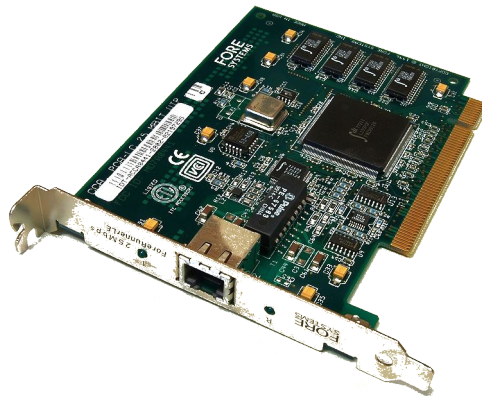


FIGURE I.2.4. – Image originale par Barcex sous licence CC BY SA 3.0

[Lien vers l'image d'origine ↗](#)

La carte réseau de la photo comporte un port femelle Ethernet : ce port peut accueillir un câble Ethernet mâle (connecteur RJ45). Les cartes réseau internes sont souvent des cartes PCI, c'est-à-dire qu'elles s'enfoncent dans un port PCI.

*i*

Une clé Wi-Fi est aussi une carte réseau à elle toute seule, sauf que contrairement à une carte comme celle ci-dessus, elle se présente sous forme d'une clé USB et se branche sur un port USB.

### 1.2.3. Concentrateur (hub)

Un *hub* est un dispositif en réseau qui permet de mettre plusieurs ordinateurs en contact. Définition pas très précise, puisque tout dispositif en réseau (ou presque) a le même but. 🍊 Bref, ce qu'il faut retenir est qu'un *hub* est très bête, enfin, moins intelligent que les autres. Ce qu'il fait est tout simple : il reçoit des données par un port, et envoie ce qu'il reçoit aux autres. Il a une interface de réception (un port) et une interface de diffusion (plusieurs autres ports par où les autres ordinateurs sont connectés).

Attention, une interface permet la réception ET la diffusion. Comme vous pouvez le voir sur la photo ci-dessous, le *hub* n'a pas juste deux interfaces physiques, où on entre par la gauche et on ressort à droite, non ! L'interface de réception est logique.

Exemple : j'ai un *hub* à 4 ports, avec 4 ordinateurs connectés. J'ai le port 1, 2, 3, 4 (ici, interface = port). Si l'ordinateur 4 (au port 4) veut communiquer avec les autres, moi le *hub*, je reçois les données au port 4 (c'est mon port de réception) et je renvoie les données aux ports 1, 2, et 3 : ce sont les ports de diffusion.



Je ne renvoie plus les données au port 4, car c'est mon port de réception. 🍊



FIGURE I.2.5. – Un hub (image domaine public)

Ce qu'on lui reproche est le manque de confidentialité. Eh oui, le *hub* ne garde pas de secret : tout ce qu'un ordinateur lui dit, il l'envoie aux autres. Heureusement, les autres vérifient bien si ça leur est destiné, et si ça ne l'est pas, ils laissent tomber les données et ne les lisent pas.



C'est toujours sécurisant, non ?

Non, pas du tout, à partir du moment où les données arrivent jusqu'à la carte réseau, elles peuvent toujours être lues. Bon, en pratique, le seul cas où on rencontre des hubs... c'est dans les cours de réseau. 🍊

## 1.2.4. Commutateur (switch) et routeur : si peu ressemblants et si similaires

Le commutateur (ou *switch*) et le routeur sont 2 appareils fondamentalement différents, et pourtant, leurs rôles se ressemblent tellement ! Au-delà de leur architecture, il faut comprendre leur différence au niveau d'un réseau.

### 1.2.4.1. Le commutateur : juste une histoire d'échange de données

Un commutateur fonctionne à peu près comme un *hub*, sauf qu'il est plus discret et intelligent. Il n'envoie pas tout ce qu'il reçoit à tout le monde, mais il l'envoie uniquement au destinataire. Si l'ordinateur 1 envoie des données à l'ordinateur 2, seul ce dernier les recevra et pas les autres connectés. Afin de déterminer l'ordinateur à qui il faut renvoyer les données, le *switch* se base sur les adresses physiques (adresses MAC) des cartes réseau. Pour faire une analogie avec la vie réelle, une adresse MAC est un peu comme une adresse postale. C'est une suite de 6 nombres hexadécimaux, par exemple 00-16-D4-C7-6E-D3. Si vous ne savez pas ce qu'est l'[hexadécimal](#) , ce n'est pas bien grave pour le moment, mais pensez à consulter [notre annexe sur le sujet](#) à l'occasion. On en a souvent besoin en informatique (et pas qu'en réseau). Nous n'étudierons pas les adresses MAC dans ce chapitre, elles seront étudiées à partir de la partie 3, lorsque nous aborderons réellement la communication dans un réseau.

Un commutateur transmet donc des données aux autres ordinateurs en se basant sur leurs adresses MAC. Les transmissions sont plus confidentielles, les autres ne savent rien des données ne leur étant pas destinées. Son utilisation reste limitée aux réseaux locaux.



FIGURE I.2.6. – Des switchs - Image par Jellyfishz sous licence CC BY SA 4.0

[Lien vers l'image d'origine](#)

### 1.2.4.2. Le routeur, un véritable ordinateur

Un routeur ressemble à un *switch* sur le plan de l'utilisation : en effet, il permet de mettre plusieurs ordinateurs en réseau. Mais cela va plus loin : il permet de mettre en contact 2 réseaux

fondamentalement différents. Dans une petite installation, avec un ou plusieurs ordinateurs connectés à une « box » (qui est en fait un routeur), il est la frontière entre le réseau local et Internet.

Un routeur a plusieurs interfaces. Pour continuer dans notre exemple de frontière avec Internet, il possède une interface connectée à Internet (généralement, cela se traduit par un câble branché sur la prise téléphonique ou sur un boîtier fibre optique) et plusieurs autres interfaces sur lesquels se connectent des ordinateurs voulant accéder à Internet (ce qui se traduit généralement par des câbles Ethernet ou des connexions Wi-Fi).



Notez aussi que le routeur n'est pas uniquement utilisé pour aller sur Internet, on l'utilise aussi dans un réseau strictement local.

### 1.2.5. Répéteur

Un répéteur (*repeater* en anglais) agit un peu comme un *hub*, mais ce dernier n'a que 2 interfaces. Son intérêt est de renvoyer ce qu'il reçoit par l'interface de réception sur l'interface d'émission, mais plus fort. On dit qu'il régénère et réémet le signal. En transmission sans fil (radio, téléphone) on parle aussi de relais. Un répéteur permet de couvrir des distances plus grandes que les distances maximales fixées par le matériel que l'on utilise : par exemple, dans un réseau sans fil (Wi-Fi), la portée maximale entre 2 appareils est d'environ 50 mètres en intérieur. En plaçant un répéteur peu avant ces 50 mètres, vous pouvez connecter 2 appareils à 100 mètres de distance. Toutefois, le fait que les informations soient renvoyées « plus fort » peut dégrader la qualité du signal dans les réseaux sans fil. Pour prendre un exemple parlant, en radiophonie, si l'on se trouve trop loin d'un relais, la qualité du son que l'on entend est dégradée.

### 1.2.6. Bilan des matériels

Afin de conclure ce chapitre, nous allons récapituler le matériel vu et son utilité. Un tableau récapitulatif vaut mieux qu'un long discours :

Matériel	Utilité
Carte réseau	La carte réseau est le matériel de base indispensable pour tout au sujet de la communication dans le réseau.
Concentrateur ( <i>hub</i> )	Le concentrateur permet de relier plusieurs ordinateurs entre eux, mais on lui reproche le manque de confidentialité.
Commutateur ( <i>switch</i> )	Le commutateur fonctionne comme le concentrateur, mais il transmet des données aux destinataires en fonction de leurs adresses MAC (adresses physiques). Une machine reçoit seulement ce qui lui est adressé.

Routeur	Le routeur permet d'assurer la communication entre différents réseaux pouvant être fondamentalement différents (réseau local et Internet).
Répéteur	Le répéteur reçoit des données par une interface et les renvoie <i>plus fort</i> par l'interface d'émision. On parle aussi de <i>relais</i> en téléphonie et radiophonie.

## Conclusion

Dans ce chapitre de culture informatique, nous avons examiné différents composants que l'on peut utiliser en réseau. Il est vraiment important de comprendre leur fonctionnement pour pouvoir choisir ce qui sera utilisé dans différents cas.

Mais on en fait quoi de tout ce matériel ? Les câbles, on les branche où ? Quelles machines doivent être reliées entre elles ?

Rendez-vous au prochain chapitre pour répondre à ces questions !

## I.3. Les topologies

### Introduction

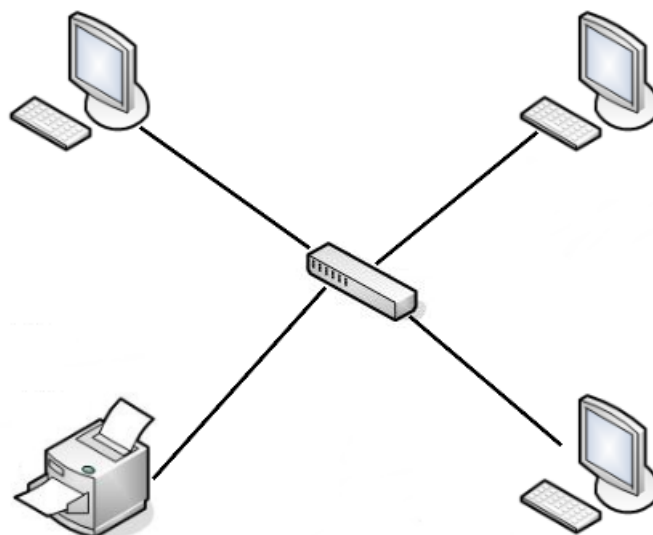
Dans ce chapitre nous allons étudier les topologies. Il s'agit des différentes formes que peuvent prendre des réseaux.

#### I.3.1. Avant tout...

Avant tout, il faut que vous connaissiez quelques types de réseaux, cela aidera à comprendre pourquoi certaines topologies existent.

##### I.3.1.1. LAN: le réseau local

Un LAN, *Local Area Network* (en français réseau local) est un réseau limité à un espace géographique comme un bâtiment. Par exemple, l'ensemble des ordinateurs dans une école forme un LAN.



Les réseaux de zéro - zestedesavoir.com

FIGURE I.3.1. – Représentation schématique d'un LAN simple



Un WLAN, *Wireless Local Area Network*, ou *Wireless LAN*, est un LAN mais qui utilise la transmission sans fil (Wi-Fi, ...). Le mot *wireless* signifie « sans fil » (*wire* = câble, *less* = sans). Par exemple, un *hotspot* Wi-Fi, c'est-à-dire un point d'accès Wi-Fi public comme on en trouve dans des lieux publics tels qu'un hôtel, est un réseau local sans fil (WLAN).

### I.3.1.2. WAN: le réseau étendu

WAN signifie *Wide Area Network*, en français, on peut le traduire par « réseau étendu ». Il s'agit d'une interconnexion de réseaux sur de longues distances. Quand une grande entreprise interconnecte ses différents sites (usines de production, plateformes logistiques, ...), on parle de WAN. Les réseaux étendus se caractérisent par l'utilisation de technologies différentes des LAN. Par exemple, alors qu'on trouve souvent des câbles Ethernet ou du Wi-Fi sur des réseaux locaux, les connexions WAN se font plutôt en fibre optique voire en 4G.

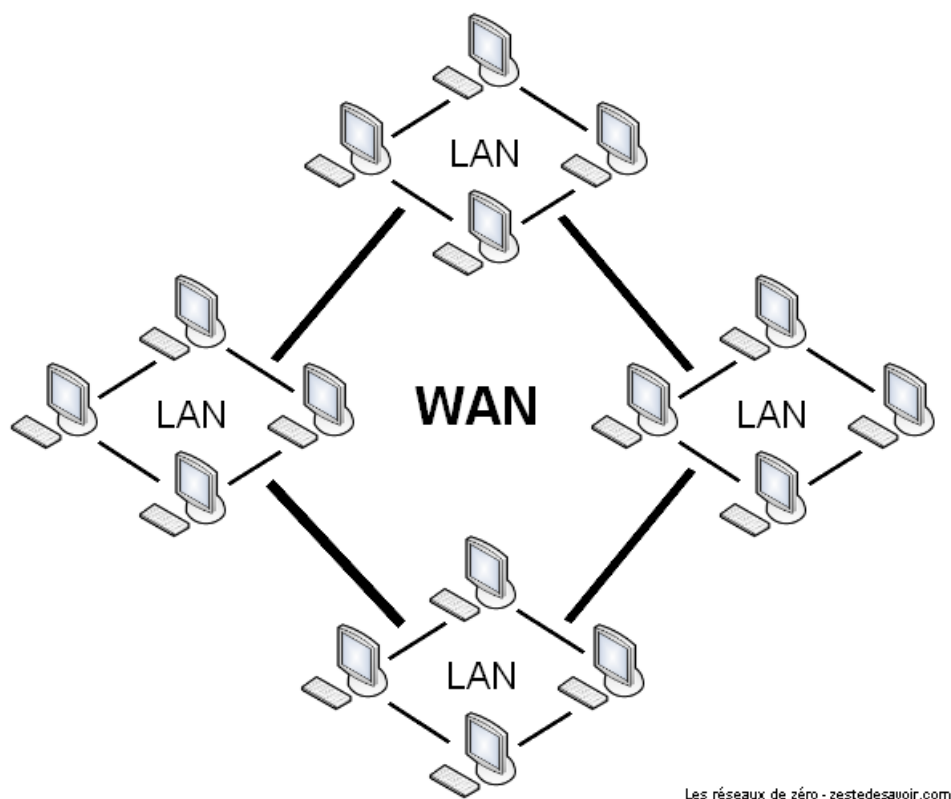


FIGURE I.3.2. – Représentation schématique d'un WAN

### I.3.2. C'est quoi une topologie ?

Bonne question, qu'est-ce qu'une topologie ? Tout d'abord, il faut savoir qu'il existe deux types de topologies : physique et logique.

### 1.3.2.1. Topologie physique

Une topologie physique est en fait la structure physique de votre réseau. C'est donc la forme, l'apparence du réseau. 🍊

Il existe plusieurs topologies physiques : le bus, l'étoile (la plus utilisée), le mesh (topologie maillée), l'anneau, hybride, etc. Cependant nous n'allons parler que des plus utilisées.

### 1.3.2.2. Topologie logique

Une topologie logique est la structure logique d'une topologie physique, c'est-à-dire que la topologie logique définit *comment* se passe la communication dans la topologie physique. 🍊



Attention avec ces deux notions !

L'une (topologie physique) définit la **structure physique** (l'apparence physique, la forme) de votre réseau, l'autre (topologie logique) définit **comment la communication se passe** dans cette forme physique. 🍊 Retenez bien ces 2 notions, et ne les confondez pas, tant qu'à faire. 🍊

### 1.3.3. Réseau en bus

Comme son nom l'indique, la topologie bus a les caractéristiques d'un bus (pensez, une ligne droite). Dans cette topologie, tous les ordinateurs sont connectés entre eux par le biais d'un seul câble réseau débuté et terminé par des **terminateurs**.

Les terminateurs ont pour but de maintenir les **frames** (signaux électriques de données) dans le câble et d'empêcher les « rebonds » des données le long du fil. 🍊

Franchement, ce n'est pas pratique du tout, et ce pour 2 raisons majeures. La première est que, parce que toutes les machines utilisent le même câble, s'il vient à ne plus fonctionner, alors le réseau n'existe plus. Il n'y a plus de communication possible étant donné que tous les hôtes partagent un câble commun. La seconde est que, puisque que le câble est commun, la vitesse de transmission est très faible. 🍊 Il y a d'autres raisons qui font que cette topologie est très peu utilisée.

Dans cette topologie, étant donné que le câble de transmission est commun, il ne faut pas que 2 machines communiquent simultanément, sinon... Bam, ça crée des collisions ! 🍊 Pour éviter ce problème, on utilise une méthode d'accès appelée CSMA/CD. Avec cette méthode, une machine qui veut communiquer écoute le réseau pour déterminer si une autre machine est en train d'émettre. Si c'est le cas, elle attend que l'émission soit terminée pour commencer sa communication. Sinon, elle peut communiquer tout de suite.

C'est un peu complexe, heureusement que d'autres topologies plus simples et plus pratiques existent !

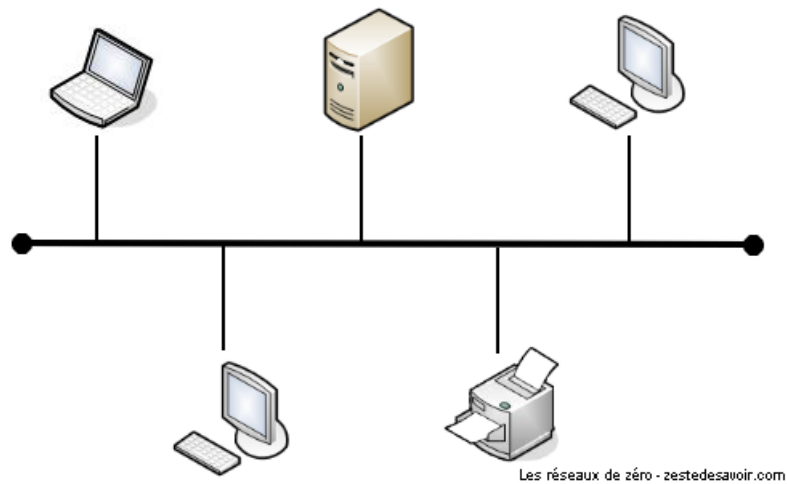


FIGURE I.3.3. – Représentation schématique d'un réseau en bus

### I.3.4. Topologie de type étoile

Dans un réseau en étoile, la forme physique du réseau ressemble à une étoile. Une image est plus parlante :

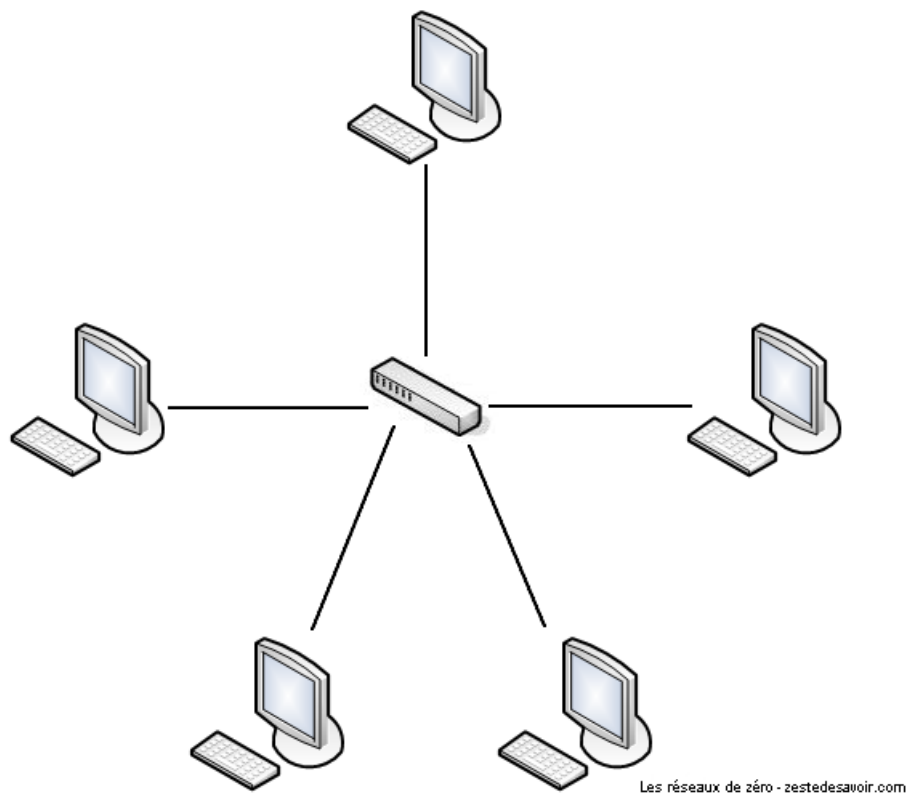


FIGURE I.3.4. – La forme physique du réseau ressemble à une étoile

N'importe quel appareil (routeur, commutateur, concentrateur, ...) peut être au centre d'un réseau en étoile. L'important, c'est que pour parler à une autre entité on passe par le matériel

## I. Le concept et les bases

central (qui peut être le *hub*, le *switch*, etc.). En pratique, dans un réseau d'entreprise en étoile, au centre on trouve un *switch*. 🍊

Le principal défaut de cette topologie, c'est que si l'élément central ne fonctionne plus, plus rien ne fonctionne : toute communication est impossible. Cependant, il n'y a pas de risque de collision de données.



Si vous reliez des ordinateurs à un *hub*, la topologie **physique** sera l'étoile. Mais la topologie **logique** sera... le bus ! En effet, sur un *hub*, seule une machine peut émettre à la fois. Les autres doivent écouter le réseau pour savoir si elles peuvent émettre !

### 1.3.5. Réseau en anneau : le ring, mais pas de boxe

Oui bon, le jeu de mot est pourri... Enfin, vous devez commencer à avoir l'habitude ! 🍊

On attaque un morceau assez compliqué, du moins plus complexe que ce qu'on a vu jusqu'à présent. Nous allons donc essayer de faire simple (très contradictoire 🍊).

Comme vous pouvez vous en douter, un réseau en anneau a la forme d'un... anneau, oui, il n'y a pas de piège ! Cependant, la topologie physique d'un réseau en anneau est... le bus.



Mais alors un réseau en anneau c'est comme un réseau en bus avec les machines disposées en cercle ? 🍊

Si on veut, mais il a une particularité : la topologie logique est le *token ring*.



Anneau à jeton ? On met un jeton dans la machine pour avoir un anneau ? 🍊

Pas du tout ! 🍊 Rappelez-vous, la topologie de type bus possédait un problème de collision de données : 2 machines ne doivent pas échanger des données en même temps, sinon elles s'entrechoquent. Ce principe est repris dans le réseau en anneau. Sauf que là, le système de *token ring* utilise la CSMA-CA, une méthode anti-collision différente.

Le principe est assez simple : une machine connectée au réseau possède un jeton virtuel. Ce jeton, c'est une **autorisation de communiquer**. Une fois que la machine a transmis ce qu'elle voulait, elle passe le jeton à la machine suivante, et ainsi de suite. Si le détenteur du jeton n'a rien à dire, il le passe au suivant.



Vous allez nous dire qu'on radote, mais on le répète quand même : la topologie physique, ici le bus, définit la forme physique du réseau (bon ici le bus est un peu courbé... 🍊). La topologie logique, ici le *token ring*, définit la manière de communiquer dans un réseau. Si vous confondez, vous allez vous retrouver à vouloir brancher un jeton de casino sur une machine pour qu'elle puisse communiquer... 🍊

Voici une animation décrivant de manière simplifiée le fonctionnement logique d'un réseau en anneau. Le jeton rouge se transmet de machine en machine.

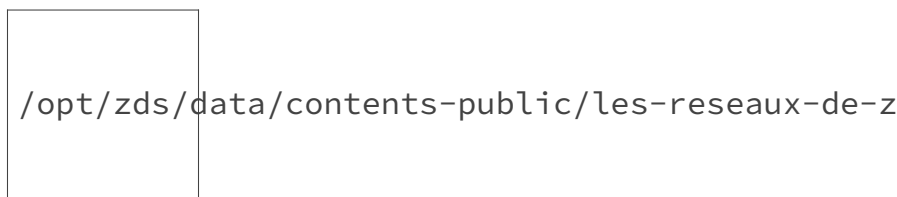


FIGURE I.3.5. – Réseau en anneau. Des ordinateurs attendent le jeton (token) pour transmettre des données.

### I.3.6. Topologie maillée

La topologie maillée est LA topologie que l'on vous souhaite de ne jamais utiliser ! 🍊

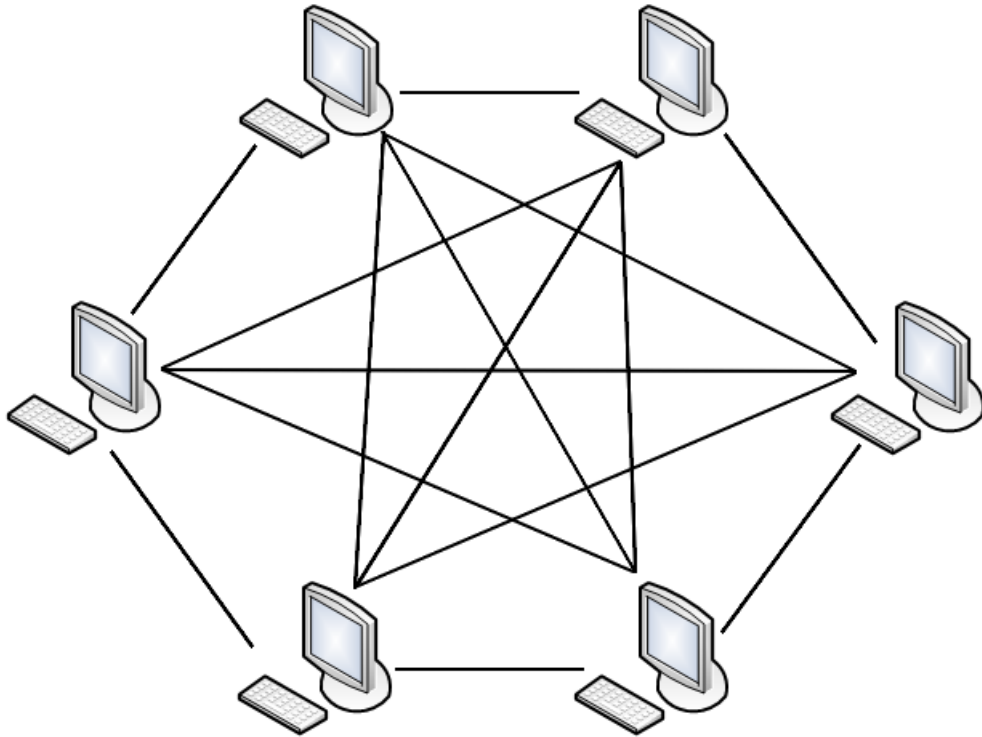
?

Pourquoi ?

Eh bien, c'est qu'il y a vraiment, vraiment, vraiment, vraiment... trop de câbles. 🍊 Le principe de la topologie maillée est de relier tous les ordinateurs entre eux (full mesh, maillage complet) ou du moins, un grand nombre. Comme ça, aucun risque de panne générale si une machine tombe en rade, mais si vous vous prenez les pieds dans des câbles, étant donné qu'il y en a partout, c'est la cata, vous faites tout tomber ! 🍊

Si on veut relier toutes les machines entre elles, la formule pour connaître le nombre de câbles est  $n(n-1)/2$ , avec  $n$  le nombre d'ordinateurs. Donc rien qu'avec 8 ordinateurs par exemple, ça nous donnera  $8(8-1)/2$ , soit jusqu'à **28 câbles** ! 🍊 En pratique, on n'a quasiment jamais recours à des réseaux *full mesh*, mais seulement partiellement maillés.

Cette topologie reste peu utilisée vu la difficulté à mettre en place une telle infrastructure. Histoire de vous faire halluciner, imaginez une école, où il y a 500 ordinateurs, si on voulait les relier tous entre eux. Ça ferait...  $500 \times (500-1)/2 = \dots$  Faites le calcul vous-même si vous ne nous croyez pas, mais ça fait bien **124.750 câbles** ! 🍊 Il ne vaut mieux même pas penser au prix que peut coûter une centaine de milliers de câbles. En plus, chaque câble doit être relié à 2 cartes réseau, ça ferait 499 cartes réseau par machine, soit 249.500 cartes réseau en tout... Donc oui, ce n'est pas facile à mettre en place, et c'est utilisé sur de petits réseaux dans des cas bien précis. 🍊



Les réseaux de zéro - zestedesavoir.com

FIGURE I.3.6. – Représentation schématisée d'un réseau complètement maillé

### I.3.7. Topologie hybride

Une topologie hybride, c'est très simple (enfin, dans le principe) : c'est juste le regroupement de plusieurs topologies différentes. Par exemple, Internet est une parfaite illustration d'un réseau hybride car il joint des réseaux en anneau avec des réseaux en bus, avec des réseaux en étoile, ... Rien de spécial au final. 🍏

i

Internet peut aussi être vu comme un réseau maillé, dans son sens logique. Rappelez-vous, dans un réseau maillé, la multiplication du nombre de câbles permet plusieurs chemins de communication (dans le réseau Internet, toutes les machines ne sont pas toutes reliées entre elles, c'est un mélange de regroupements de nœuds et autres joyeusetés). 🍏 Comme il y a tout plein de câbles, il y a donc plusieurs chemins possibles pour parler à son destinataire. On peut donc décrire Internet comme un réseau maillé (dans le sens logique), car on peut transmettre des données par plusieurs chemins.

## Conclusion

Il faut avouer que ce chapitre n'était pas vraiment difficile. Ce qu'il faut comprendre et maîtriser, c'est la différence entre une topologie physique et une topologie logique. Dans le monde professionnel, on utilise généralement des topologies (physiques et logiques) de type

## *I. Le concept et les bases*

étoile. Vous êtes tout de même susceptible de tomber sur des infrastructures anciennes qui reposent sur d'autres topologies, car il existe une règle implicite avec les systèmes informatiques dans l'industrie : **tant que ça marche, on n'y touche pas** ! Ce n'est pas un conseil, c'est un constat. 🍊

Maintenant qu'on a fait un rapide tour du matériel, il faudrait peut-être établir des communications ! Pour cela, direction la partie 2, où on va se pencher sur les **protocoles** et sur le **modèle OSI** ! 🍊

# Conclusion

Maintenant que vous connaissez la théorie nécessaire, nous allons pouvoir passer à la seconde partie.

## Deuxième partie

Un modèle qui en tient une couche

# Introduction

Quittons le matériel pour nous intéresser aux fondements de la communication. On va voir ce que sont les protocoles, quels sont ceux qu'on utilise couramment, puis on va commencer à rentrer dans des choses un peu moins évidentes mais absolument passionnantes : le modèle en couches. 🍊

## II.1. Introduction aux protocoles

### Introduction

Pour que des machines puissent communiquer entre elles, elles doivent respecter certains **protocoles**. Mais qu'est-ce qu'un protocole ?

#### II.1.1. Vous avez dit protocole ?

Avant d'étudier comment communiquent les hôtes dans un vaste réseau tel qu'Internet, il nous faut comprendre ce qu'est un protocole pour commencer.

■ A protocol is a set of rules that define how communication occurs in a network.

C'est la définition la plus basique d'un protocole que vous retrouverez certainement dans plusieurs cours anglais de réseaux. En français, on dit qu'un protocole est **un ensemble de règles qui définissent comment se produit une communication dans un réseau**. Pour mieux appréhender cela, nous allons considérer une analogie.

##### II.1.1.1. Le protocole : un genre de langue

Communiquer est l'une des activités les plus courantes. Les personnes qui communiquent ne peuvent se comprendre que dans deux cas :

- Si elles parlent la même langue.
- Si elles ont un intermédiaire qui parle leurs deux langues respectives pour faire office d'interprète.

Mais une langue que les humains parlent, qu'est-ce que c'est, au final ?

D'après Wikipédia, une langue est un système constitué de signes linguistiques, vocaux, graphiques, gestuels, tenu en cohésion par des règles précises qui, lorsque respectées, permettent la communication.

En réseau, c'est la même chose. La langue que les humains parlent, c'est un protocole pour les hôtes dans un réseau. Pas n'importe quel protocole, car il en existe plusieurs. Mais celui qui nous concerne est appelé « **protocole de communication** ».

Quant à l'interprète de notre exemple, dans un réseau, ce sera la passerelle (applicative) qui permettra de faire communiquer deux réseaux basés sur des protocoles différents en assurant plusieurs fonctions telles que la traduction des protocoles et des signaux, l'isolation d'erreurs, l'adaptation d'impédances, etc.

## II. Un modèle qui en tient une couche

Si vous ne comprenez pas ces termes techniques, ce n'est pas important pour l'instant.

### II.1.2. L'utilité d'un protocole par l'exemple

Bien ! Vous avez compris le concept de protocole ? Maintenant essayons de voir à quoi ça sert dans un réseau. Pour comprendre cela, très souvent, on utilise une analogie que nous qualifions de « classique » en réseau, car plusieurs professeurs utilisent presque toujours cette dernière pour faire assimiler les fonctions assurées par un protocole. Il s'agit de la communication téléphonique entre deux humains.

Pierre veut transmettre un message à Jean. Il compose donc son numéro de téléphone et il peut entendre la tonalité (tuuuut... tuuuut...). Il attend que Jean décroche, car la communication ne peut avoir lieu qu'à ce moment-là. Jean, de son côté, entend son téléphone sonner. Il décroche, et c'est là qu'intervient le classique « Allô ? ». 🍊

À ce niveau, la « session de communication » est établie, c'est-à-dire que Pierre peut maintenant dire à Jean ce qu'il a en tête. Il va donc gentiment se présenter : « **Salut, c'est Pierre...** » et évoquer le contexte ou la raison de son appel : « **C'était juste pour te dire que demain, il y aura une fête chez Anne-Sophie, qui habite au numéro 10 de la rue Lézard !** ». Jean peut éventuellement demander à Pierre de répéter, pour être sûr d'avoir bien saisi son message : « **Chez qui ? Anne qui ?** ». Alors Pierre répétera cette partie pour que Jean comprenne. Finalement, la conversation terminée, il faut se séparer en douceur ( 🍊 ). Un classique « salut » ou « au revoir » des deux côtés avant qu'ils ne raccrochent leurs combinés.

Les protocoles nous permettent de faire tout ça. Essayons un peu de réexaminer ce scénario avec un langage un peu plus informatique. 🍊

Pierre veut transmettre un message à Jean.

Il compose donc son numéro de téléphone et il peut entendre la tonalité (tuuuut... tuuuut...).

Il attend que Jean décroche, car la communication ne peut avoir lieu qu'à ce moment-là.

L'hôte Pierre, à l'adresse IP 124.23.42.13, souhaite communiquer avec l'hôte Jean à l'adresse IP 124.23.12.13. Il lui enverra un paquet de demande d'initialisation de session (il compose son numéro et attend que Jean décroche et dise « Allô »). À ce stade, il peut se passer quatre choses dans le contexte naturel :

1. Le numéro est incorrect.
2. Le numéro est correct mais indisponible.
3. Le numéro est correct et Jean décroche en disant « Allô ».
4. Le numéro est correct, disponible, mais Jean ne décroche pas (c'est donc un peu comme le cas 2 🍊 ).

Étudions ces cas :

- Cas 1 : Pierre aura un message vocal disant « **Le numéro que vous avez composé n'existe pas** ». En réseau ce sera un paquet **ICMP**(*Internet Control Message Protocol*) qui enverra **une erreur de type 3 (destination unreachable, destination inaccessible)** et de code **7 (destination host unknown, destinataire inconnu)**.



ICMP est un protocole dans la suite protocolaire TCP-IP utilisé pour envoyer des messages d'erreurs dans un réseau. Il travaille en partenariat avec le protocole IP. Nous allons le voir en détail, voir les différents types d'erreurs, leurs codes, leurs significations et les scénarios dans lesquels elles se manifestent. 🍊

- Cas 2 : Ici, un message vocal dira à Pierre « L'abonné que vous souhaitez appeler est injoignable pour l'instant, veuillez rappeler dans quelques instants ». En réseau, il s'agira également d'une erreur de type 3.
- Cas 3 : Si le numéro est correct et que Jean décroche en disant « Allô », c'est le début de la conversation. En réseau on dira donc qu'une session a été initialisée. 🍊
- Cas 4 : Ici, classiquement, ce sera le répondeur de Jean qui dira « **Je ne suis pas disponible pour l'instant, laissez-moi un message, je vous rappellerai dès que possible** ». En réseau, c'est un peu différent. L'hôte Pierre va recevoir **une erreur ICMP de type 3 (destination inaccessible) et de code 1 (destinataire inaccessible)**. En gros, c'est pour dire qu'on n'arrive pas à atteindre le destinataire. En fait, si un numéro de téléphone est disponible, sonne, mais que personne ne répond, ça veut dire qu'on n'a pas atteint le destinataire final en fait. Donc c'est un peu pareil que le cas 2.

Continuons l'analyse de notre analogie. 🍊

« C'était juste pour te dire que demain, il y aura une fête chez Anne-Sophie, qui habite au numéro 10 de la rue Lézard ».

Jean peut éventuellement demander à Pierre de répéter, pour être sûr d'avoir bien saisi son message « Chez qui ? Anne qui ? ». Alors Pierre répétera cette partie pour que Jean comprenne.

Si Jean demande à Pierre de répéter quelque chose, de façon radicale on peut conclure qu'il n'a pas **reçu** ce que Pierre a dit (si l'on considère que recevoir un message = comprendre le message). En réseau, l'hôte Jean va envoyer un paquet à Pierre disant « **je n'ai pas reçu le dernier paquet, renvoie-le stp** ». Pierre va alors renvoyer le dernier paquet. En fait, c'est un peu plus précis que ça. Suivant le protocole que vous utilisez (UDP ou TCP, nous allons les comparer dans les prochains chapitres), Pierre peut demander à la fin de chaque phrase si Jean a compris. En réseau, l'hôte Pierre pourrait donc demander un message d'accusé de réception à chaque envoi de paquet, et l'hôte Jean devra répondre « **oui j'ai reçu, envoie le prochain** » tout le long de la communication si l'on utilise le protocole TCP qui est dit **connection-oriented (orienté connexion)** par opposition au protocole UDP qui est dit **connectionless-oriented**. Tenez-vous tranquille, avec TCP on peut faire encore plus fort que ça.

Qu'est-ce qui se passe, si Pierre se met à ~~raconter sa vie~~ à raconter une histoire à Jean et que ce dernier dépose le combiné et s'en va faire un tour aux toilettes sans prévenir ? Pierre aurait perdu son temps en parlant pour rien ! Pour prévenir ce genre de chose, Pierre peut vérifier la présence de Jean en demandant toutes les x minutes « **Tu me suis ? Tu es là ?** ». En réseau, avec TCP il s'agit d'une **vérification périodique de l'état de la session de communication**. Ceci dit, l'hôte Pierre enverra un paquet de « **vérification de session** » pour savoir si l'hôte Jean est toujours connecté. Si Jean ne répond pas après un certain laps de temps, la communication est terminée (la session se termine là).



Ici, nous sommes dans l'explication de ce que fait le protocole TCP. Vous n'étiez pas censé le savoir, c'était juste pour vous illustrer le fonctionnement des protocoles sans vous dire duquel il s'agissait. Mais nous avons préféré vous le dire, car nous faisons allusion à des paquets ici, mais en fait il s'agit des valeurs précises qui se trouvent dans l'en-tête des paquets TCP. 🍊

Finalement, la conversation terminée, il faut se séparer en douceur. 🍊 Un classique « salut » ou « au revoir » des deux côtés avant qu'ils ne raccrochent leurs combinés.

À ce stade, la session de communication est terminée.

### II.1.3. Les exigences d'un protocole

Un protocole de communication digne de ce nom doit remplir quelques exigences rigoureuses. Un protocole est un ensemble de règles dictant comment doit s'effectuer la communication entre deux entités. Ceci dit, il faudrait que ledit protocole soit en mesure d'assurer des fonctions vitales au bon déroulement d'une communication. Il existe plusieurs « fonctions vitales » (comprendre exigences) qu'un protocole de communication doit être capable de remplir. Dans la sous-partie précédente, nous avons vu quelques-unes de ces fonctions le long de l'exemple sans vous les pointer directement. Parmi ces fonctions figurent en bonne et auguste posture :

- **La gestion du format des données** : un protocole, comme nous l'avons répété, définit comment s'effectue la communication. Or, qui dit communication dit échanges de données. Le protocole doit donc avoir des « fonctions » permettant de gérer le format de ces données. Nous verrons plus tard dans quelle couche du modèle OSI on trouve ces services de formatage. En général, les données seront constituées de deux choses : d'un entête et du contenu. L'entête sera un peu « réservé » au protocole. C'est à ce niveau que l'on trouve des informations « techniques » tandis que le contenu... bah, c'est le contenu ! 🍊
- **La gestion du format d'adresses** : durant la procédure de transmission des données, il faudrait bien gérer les adresses : qui est l'émetteur, qui est le destinataire ? Dans une communication dans le monde naturel, quand on écrit une lettre, dans l'entête, on met l'adresse de l'émetteur et celle du destinataire, et même sur l'enveloppe d'ailleurs. Si on ne le fait pas, on ne sait pas à qui envoyer la lettre, et celui qui la reçoit ne sait même pas si elle lui est destinée et de qui elle provient. Par comparaison, dans l'entête des données « encapsulées », il faudrait qu'un protocole soit en mesure de spécifier l'adresse de l'émetteur et du destinataire.
- **Correspondance d'adresses** : quand vous inscrivez l'adresse du destinataire sur une enveloppe, cette dernière est « logique ». Logique dans ce sens que le destinataire n'habite pas sur cette enveloppe ( 🍊 ), mais cette adresse indique l'adresse physique du destinataire, là où vous pouvez le trouver si vous vous y rendez physiquement. 🍊 Le facteur doit donc faire une correspondance entre cette adresse logique sur l'enveloppe et l'adresse physique. Par analogie, un protocole doit assurer des fonctions de correspondance entre les adresses logiques ( IP ) et les adresses physiques ( MAC ). Cette correspondance s'appelle « *address mapping* » en anglais. 🍊

## II. Un modèle qui en tient une couche

- **Routage** : nous allons passer un long moment sur ce sujet dans la suite de ce tuto. Dit simplement, le routage consiste à « diriger » les données entre deux réseaux d'un plan d'adressage différent.
- **Détection d'erreurs de transmission** : il se peut qu'une erreur se produise dans la procédure de transmission des informations. Un protocole devrait donc être en mesure de détecter ces erreurs. Comme nous allons le voir, il s'agit d'un **CRC (Cyclic Redundancy Check, Contrôle de Redondance Cyclique)** qui est ajouté à la fin des paquets.
- **Accusé de réception** : quand vous recevez un mail, très souvent vous y répondez. Cette réponse informe explicitement à l'émetteur que vous avez reçu son mail. C'est en quelque sorte un accusé de réception. Certains protocoles permettent donc à un hôte récepteur d'informer un hôte émetteur qu'il a reçu le paquet envoyé pour empêcher ce dernier de renvoyer les mêmes choses. D'autres par contre n'implémentent pas cette fonction.
- **La gestion de perte d'informations** : de même que des erreurs peuvent se produire lors de la transmission, il peut y avoir des pertes d'informations. Pertes ? Dans un réseau ? Oui ! Généralement quand un paquet met trop du temps à arriver à son destinataire, « **il se perd** ». 🍊 Voilà pourquoi c'est important qu'un protocole gère la reconnaissance des paquets. Si l'hôte-récepteur B répond dans un intervalle de x secondes à l'hôte-émetteur A, ce dernier saura alors que B a bien reçu les données, et n'essayera plus de les renvoyer. Si B par contre ne répond pas à A, ce dernier peut donc conclure que les données « **se sont perdues** » et va les renvoyer dans un espace de temps déterminé par le protocole.
- **La direction du flux d'informations** : A et B peuvent-ils communiquer (s'échanger des données) simultanément ? Si oui, il s'agit d'un système de communication **full-duplex**. Sinon, il s'agit d'un système de communication **half-duplex**. Nous allons en parler un peu plus tard dans cette partie du cours. 🍊 Un protocole doit donc dicter la direction de flux dans la communication pour empêcher à deux hôtes de communiquer simultanément dans un système *half-duplex* par exemple.
- **Contrôle de séquences** : toute information envoyée sur un réseau est segmentée en plusieurs « séquences » (nous y reviendrons). Elles sont ensuite envoyées au destinataire. Selon la congestion (le degré d'occupation) des routes qu'elles vont emprunter, elles peuvent arriver « en désordre », ou même en double (s'il y a eu des retransmissions). Grâce au contrôle de séquences d'un protocole, on peut « numéroter » chaque « morceau » afin que le destinataire sache les « remettre en ordre » ou supprimer les doublons. Nous allons voir comment fonctionne cette « segmentation » en étudiant le protocole **BitTorrent**.
- **Gestion de flux** : quand deux personnes parlent, il est nécessaire de donner à celui qui « écoute » le temps de comprendre ce qui est dit, puisqu'il se peut que l'émetteur parle plus vite que le récepteur. Il faut donc gérer cette volubilité, ce flux. Dans les réseaux, il y a des cas où un hôte-émetteur peut transmettre plus vite que ne peut recevoir un hôte-récepteur. C'est là qu'intervient l'utilité de la gestion des flux.



Un seul protocole peut faire tout ça ? 🦉

Mais non ! 🍊 Les fonctions citées ne peuvent pas être réalisées par un seul protocole. Il s'agit d'une suite protocolaire, une suite de protocoles. Il y a des protocoles qui s'occupent de la transmission, d'autres du routage, etc. Une suite de protocoles est un ensemble de protocoles fonctionnant en harmonie et cohésion pour le bon déroulement de la communication. Vous avez

## II. Un modèle qui en tient une couche

déjà entendu l'expression « **protocole TCP/ IP** » ? En fait, ce n'est pas un protocole. **TCP** en est un, **IP** en est un autre. Mais **TCP/ IP**, ça fait deux. 🍊 C'est une suite (une **pile** pour être précis) de protocoles en fait, ***protocol stack*** en anglais. 🍊

## Conclusion

Voilà, les bases sont posées ! Vous savez ce qu'est un protocole, maintenant. Tout au long du cours, nous allons parler des protocoles les plus courants et importants. Mais avant cela, nous allons survoler un peu le modèle OSI, qui est à la base de la plupart des communications informatiques.

## II.2. Ils en tiennent une couche : OSI et TCP/IP

### Introduction

Ne soyez pas déçu · e ! Nous n'en sommes qu'au début du cours, alors ce chapitre sera plus une introduction aux modèles que *Le Modèle OSI de A à Z en vingt-cinq volumes...* Mais ne vous inquiétez pas, vous aurez déjà fort à faire ! 🐱

#### II.2.1. Le modèle OSI en douceur

Dans cette sous-partie, nous allons définir le plus simplement possible ce qu'est le modèle OSI. En effet, vous allez le comprendre, il n'y a aucun rapport avec la mode ni la 3D (si si, nous vous le jurons).

##### II.2.1.1. Qu'est-ce que le modèle OSI ?

Le modèle OSI (*Open Systems Interconnection* : « interconnexion de systèmes ouverts ») est une façon standardisée de segmenter en plusieurs blocs le processus de communication entre deux entités. Chaque bloc résultant de cette segmentation est appelé couche. Une couche est un ensemble de services accomplissant un but précis. La beauté de cette segmentation, c'est que chaque couche du modèle OSI communique avec la couche au-dessus et au-dessous d'elle (on parle également de couches adjacentes). La couche au-dessous pourvoit des services que la couche en cours utilise, et la couche en cours pourvoit des services dont la couche au-dessus d'elle aura besoin pour assurer son rôle. Voici un schéma pour illustrer ce principe de communication entre couches :

## II. Un modèle qui en tient une couche

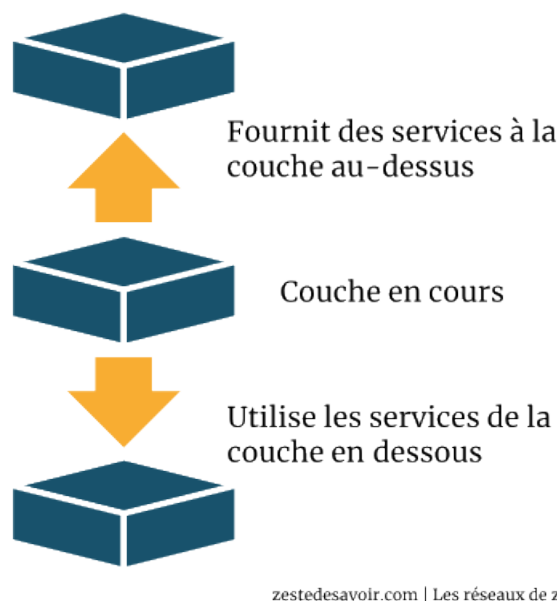


FIGURE II.2.1. – Représentation schématique d'un modèle en couches (CC BY)

Ainsi le modèle OSI permet de comprendre de façon détaillée comment s'effectue la communication entre un ordinateur A et un ordinateur B. En effet, il se passe beaucoup de choses dans les coulisses entre l'instant  $t$ , où vous avez envoyé un mail (par exemple), et l'instant  $t1$ , où le destinataire le reçoit.

Le modèle OSI a segmenté la communication en sept couches :

- Application (ou couche applicative).
- Présentation.
- Session.
- Transport.
- Réseau.
- Liaison de données.
- Physique.

Une façon efficace de connaître ces couches par cœur, de haut en bas et en anglais, serait de mémoriser la phrase suivante : **All People Seem To Need Data Processing**, ce qui signifie : « Tout le monde a besoin du traitement de données. » Chaque majuscule représente la première lettre d'une couche : A pour application, P pour présentation, S pour session, T pour transport, N pour réseau (*network* en anglais), D pour *data* (liaison de données) et finalement le dernier P (*processing*) pour physique.

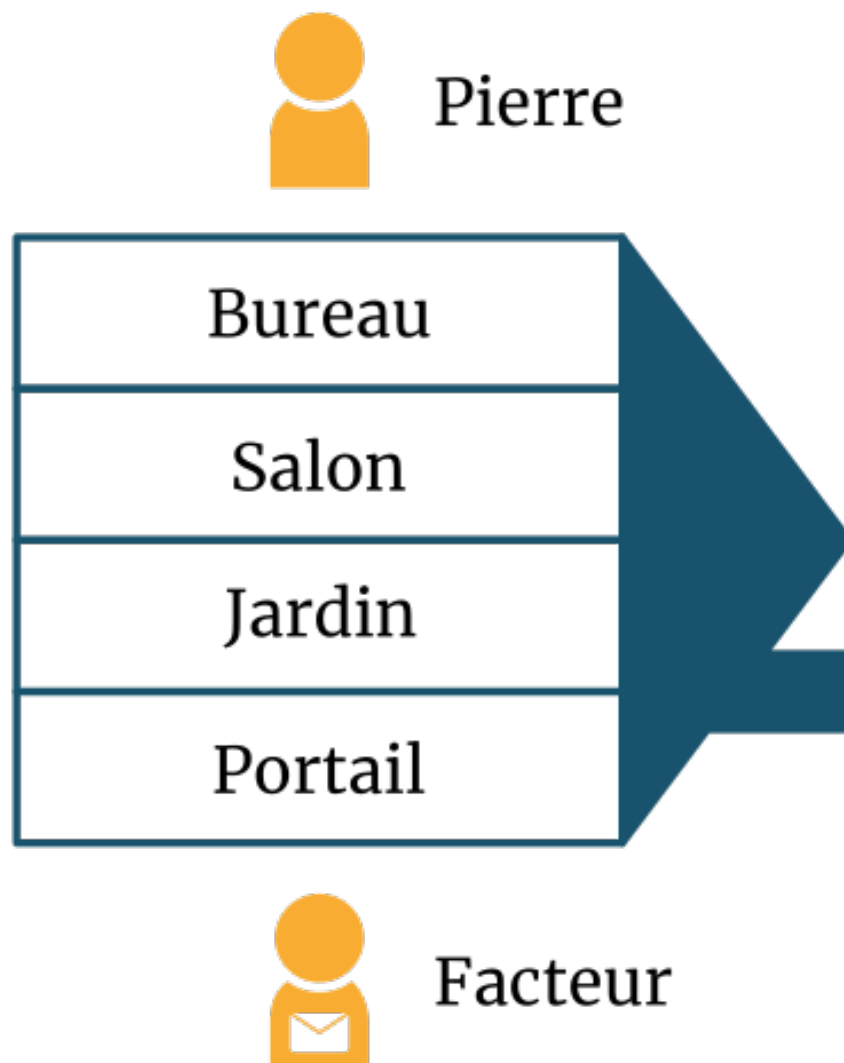
De bas en haut, le moyen mnémotechnique anglais utilisé est **Please Do Not Throw Sausage Pizza Away**. Ce qui donne en français : « S'il vous plaît, ne jetez pas les saucisses de pizza. » Ces sacrés anglophones ont des inspirations hilarantes ! 🍕 On trouve d'autres moyens mnémotechniques en français pour retenir ces couches de bas en haut : **Partout Le Roi Trouve Sa Place Assise**, ou encore : **Pour Le Réseau, Tout Se Passe Automatiquement**.

## II.2.2. Le modèle OSI par l'exemple : le facteur

Oui, nous le savons, vous êtes impatient · e ; néanmoins, allons-y lentement mais sûrement. 🍊 Nous n'allons rien vous enseigner de trop complexe, rassurez-vous. Nous avons pris l'habitude de toujours illustrer nos propos par un exemple concret, une analogie parlante.

Pour comprendre le modèle OSI, nous allons inventer un scénario. Vous vous souvenez de Pierre et de Jacques ? Oui, nos camarades d'antan ! Pierre garde une lettre dans son bureau. Il veut la donner au facteur, qui attend devant le portail de sa belle villa. La lettre est destinée à Jacques, mais Pierre n'a pas le droit d'entrer dans le bureau de Jacques. Jacques non plus n'a pas le droit de sortir de son bureau. Seul le facteur peut entrer dans le bureau de Jacques pour délivrer la lettre, mais il lui est interdit d'aller dans celui de Pierre pour la chercher.

La maison de Pierre est mal construite : il n'y a pas de couloir, juste un alignement vertical de pièces séparées par une porte. Pour aller du bureau au portail, Pierre doit traverser le salon et le jardin. Schématiquement, cela donne ceci :



zestedesavoir.com | Les réseaux de zéro

FIGURE II.2.2. – Une maison presque réaliste (CC BY)

## II. Un modèle qui en tient une couche

Dans le schéma ci-dessus, chaque pièce de la maison peut être considérée comme une couche. Pierre doit quitter la couche la plus élevée pour se diriger vers la plus basse (le portail). Une fois la lettre remise au facteur, ce dernier devra faire l'inverse chez Jacques, c'est-à-dire quitter la couche la plus basse pour aller vers la couche la plus élevée (le bureau de Jacques).

Chaque pièce de la maison possède une fonction précise. Le bureau est généralement réservé au travail ; le salon, à la distraction (discussions, télévision, etc.). Le jardin, lui, nous offre sa beauté et son air pur. Quant au portail, il permet d'accéder aussi bien au jardin qu'à la maison.

Faisons intervenir un autre personnage, Éric, dans notre histoire. Éric ne connaît absolument rien au processus de transfert de lettres. Alors quand Pierre lui dit : « J'ai écrit une lettre à Jacques », Éric imagine le scénario suivant :

- Pierre a écrit la lettre.
- Il l'a envoyée.
- Jacques a reçu la lettre.

Éric, c'est un peu vous avant la lecture de ce tutoriel. 🍊 Vous pensiez sans doute qu'après avoir envoyé un mail, par exemple, M. le destinataire le recevait directement. Mais vous venez de comprendre grâce à l'exemple de la lettre que votre mail est passé par plusieurs couches avant d'arriver au destinataire. Cet exemple vous semble peut-être aberrant, mais nous pensons qu'il a aidé plusieurs personnes à mieux concevoir le principe du modèle OSI.



Pour illustrer ce processus et faciliter votre compréhension, nous n'avons abordé que quelques couches du modèle OSI en faisant appel à un facteur. N'en déduisez pas quoi que ce soit !

### II.2.3. Survol des couches du modèle OSI

Nous y sommes presque ! Nous allons regarder le modèle OSI d'un œil plus technique, cela devrait vous plaire ! 🍊 Le modèle OSI est donc constitué de sept couches distinctes. Dans chacune de ces couches opèrent un certain nombre de protocoles.

#### II.2.3.1. Comment ça fonctionne ?

Lorsque vous voulez envoyer un mail à l'équipe des rédacteurs de ce tutoriel (comment ça, ça ne vous tente pas ? 🍊 ), plusieurs choses se passent en coulisses.

##### II.2.3.1.1. Couche applicative

Vous avez besoin d'accéder aux services réseaux. La couche applicative fait office d'interface pour vous donner accès à ces services, qui vous permettent notamment de transférer des fichiers, de rédiger un mail, d'établir une session à distance, de visualiser une page web... Plusieurs protocoles assurent ces services, dont FTP (pour le transfert des fichiers), Telnet (pour l'établissement des sessions à distance), SMTP (pour l'envoi d'un mail), etc.

## II. Un modèle qui en tient une couche

### II.2.3.1.2. Couche présentation

Il vous faut formater votre mail pour une bonne présentation. C'est dans la couche... présentation que cela se passe. Elle s'occupe de la sémantique, de la syntaxe, du chiffrement/déchiffrement, bref, de tout aspect « visuel » de l'information. Un des services de cette couche, entre autres : la conversion d'un fichier codé en EBCDIC (*Extended Binary Coded Decimal Interchange Code*) vers un fichier codé en ASCII (*American Standard Code for Information Interchange*).

i

Le chiffrement peut être pris en charge par une autre couche que la couche de présentation. En effet, il peut s'effectuer dans la couche application, transport, session, et même réseau. Chaque niveau de chiffrement a ses avantages.

i

Certains protocoles, tels que le HTTP, rendent la distinction entre la couche applicative et la couche de présentation ambiguë. Le HTTP, bien qu'étant un protocole de la couche applicative, comprend des fonctionnalités de présentation comme la détection du type de codage de caractères utilisé.

### II.2.3.1.3. Couche session

Une fois que vous êtes prêt(e) à envoyer le mail, il faut établir une session entre les applications qui doivent communiquer. La couche session du modèle OSI vous permet principalement d'ouvrir une session, de la gérer et de la clore. La demande d'ouverture d'une session peut échouer. Si la session est terminée, la « reconnexion » s'effectuera dans cette couche.

### II.2.3.1.4. Couche transport

Une fois la session établie, le mail doit être envoyé. La couche de transport se charge de **préparer** le mail à l'envoi. Le nom de cette couche peut prêter à confusion : elle n'est pas responsable du transport des données proprement dit, mais elle y contribue. En fait, ce sont les quatre dernières couches (transport, réseau, liaison de données et physique) qui toutes ensemble réalisent le transport des données. Cependant, chaque couche se spécialise. La couche de transport divise les données en plusieurs segments (ou séquences) et les réunit dans la couche transport de l'hôte récepteur (nous y reviendrons). Cette couche permet de choisir, en fonction des contraintes de communication, la meilleure façon d'envoyer une information. « Devrai-je m'assurer que la transmission a réussi, ou devrai-je juste l'envoyer et espérer que tout se passe bien ? Quel port devrai-je utiliser ? » La couche de transport modifie également l'en-tête des données en y ajoutant plusieurs informations, parmi lesquelles les numéros de ports de la source et de la destination. Le protocole TCP (*Transmission Control Protocol*) est le plus utilisé dans la couche de transport.

## II. Un modèle qui en tient une couche

### II.2.3.1.5. Couche réseau

Maintenant que nous savons quel numéro de port utiliser, il faut aussi préciser l'adresse IP du récepteur. La couche réseau se charge du routage (ou relai) des données du point A au point B et de l'adressage. Ici aussi, l'en-tête subit une modification. Il comprend désormais l'en-tête ajouté par la couche de transport, l'adresse IP source et l'adresse IP du destinataire. Se fait également dans cette couche le choix du mode de transport (mode connecté ou non connecté, nous y reviendrons là encore). Le protocole le plus utilisé à ce niveau est bien sûr le protocole IP.

### II.2.3.1.6. La couche liaison

Présentation effectuée ? O.K. !

Session établie ? O.K. !

Transport en cours ? O.K. !

Adresses IP précisées ? O.K. !

Il reste maintenant à établir une liaison « physique » entre les deux hôtes. Là où la couche réseau effectue une liaison logique, la couche de liaison effectue une liaison de données physique. En fait, elle transforme la couche physique en une liaison, en assurant dans certains cas la correction d'erreurs qui peuvent survenir dans la couche physique. Elle fragmente les données en plusieurs trames, qui sont envoyées une par une dans un réseau local. Par conséquent, elle doit gérer l'acquittement des trames (nous... enfin bref, ce chapitre n'est qu'une introduction, vous l'avez compris 🍊). Quelques exemples de protocoles de cette couche : Ethernet, PPP (*Point to Point Protocol*), HDLC (*High-Level Data Link Control*), etc.

*i*

La couche 2 assure la livraison des trames dans un réseau local. Cela dit, elle utilise des adresses physiques, la transmission des données au-delà du réseau local ne peut donc pas être gérée à ce niveau. Logique, quand on y pense : c'est le rôle de la couche 3. Tous les protocoles de cette couche n'ont pas forcément la possibilité de gérer l'acquittement des trames, qui se fait alors dans une couche supérieure.

### II.2.3.1.7. Finalement : la couche physique

Notre mail est en cours de transport, mettons-le sur le média. La couche physique reçoit les trames de la couche de liaison de données et les « convertit » en une succession de *bits* qui sont ensuite mis sur le média pour l'envoi. Cette couche se charge donc de la transmission des signaux électriques ou optiques entre les hôtes en communication. On y trouve des services tels que la détection de collisions, le *multiplexing*, la modulation, le *circuit switching*, etc.

## II. Un modèle qui en tient une couche

### II.2.3.2. Résumé

Nous avons abordé, en donnant quelques détails, chacune des couches du modèle OSI ; voici un tableau récapitulatif.

Position dans le modèle OSI	Nom de la couche	
7	Application	P
6	Présentation	E d
5	Session	R g
4	Transport	C d p le fr se
3	Réseau	C te d
2	Liaison de données	É h
1	Physique	C si

### II.2.3.3. Processus de transmission/réception

Quand un hôte A envoie un message à un hôte B, le processus d'envoi va de la couche 7 (application) à la couche 1 (physique). En revanche, quand il s'agit de recevoir, le message emprunte le chemin inverse : il part de la couche 1 (physique) pour arriver à la couche 7 (application). Souvenez-vous de l'exemple de Pierre, Jacques et le facteur : Pierre quittait le salon pour le portail afin d'envoyer sa lettre, alors que le facteur quittait le portail et se dirigeait vers le bureau de Jacques pour la délivrer.

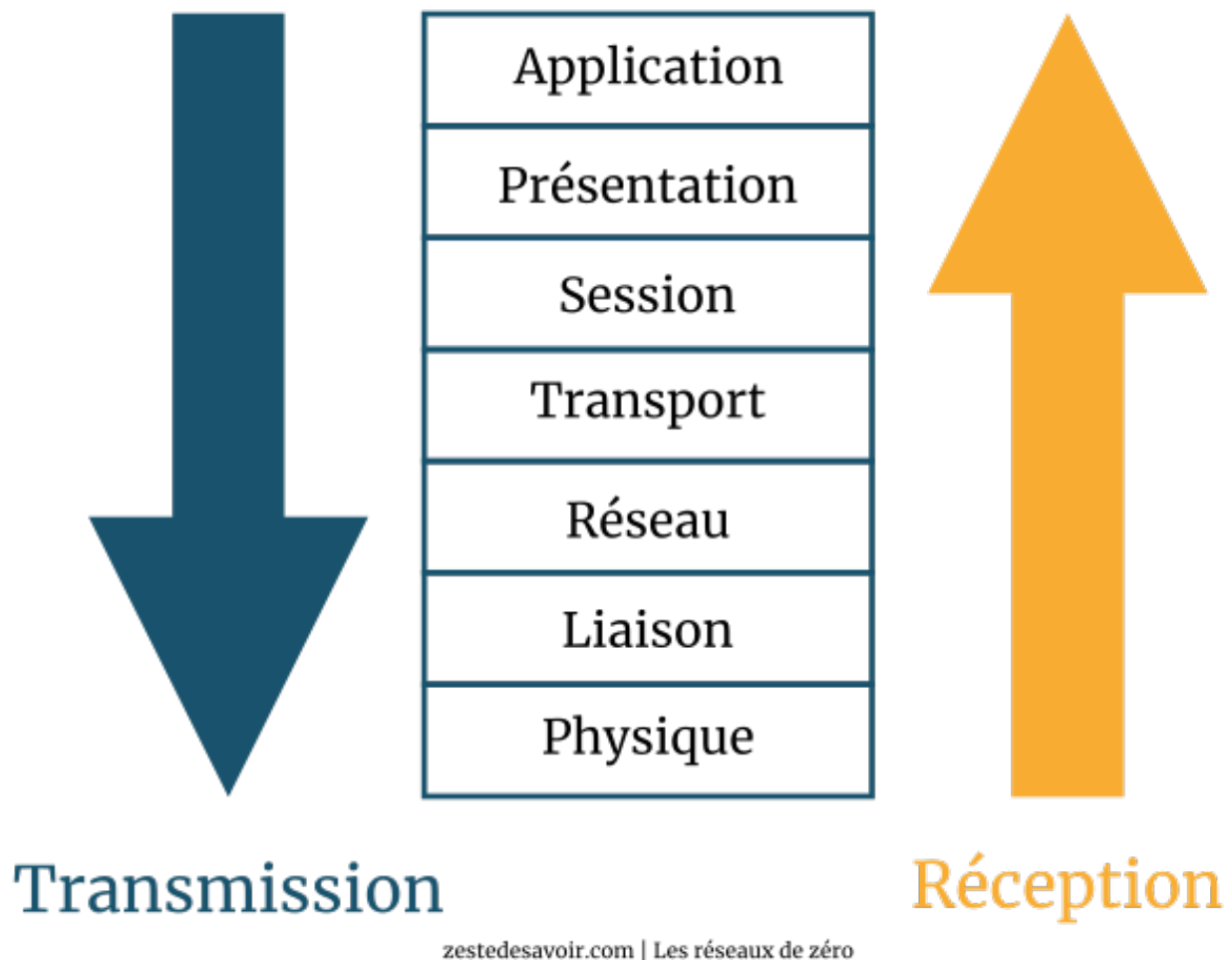


FIGURE II.2.3. – Traversée des couches pour la transmission et la réception (CC BY)

## II.2.4. TCP/IP vs OSI : le verdict ?

Vous vous êtes peut-être posé la question de savoir pourquoi le titre de ce chapitre était *Ils en tiennent une couche : OSI et TCP/IP*, alors que ce dernier semble être passé aux oubliettes. Nous allons bien étudier deux modèles différents : TCP/IP et OSI. Nous allons commencer par revoir leurs origines et le but de leur création, ensuite nous comparerons leurs architectures respectives.

### II.2.4.1. Il y a une génération...

Le modèle TCP/IP fut créé dans les années 1970 par le département de la Défense des États-Unis d'Amérique, plus précisément par l'agence DARPA (*Defense Advanced Research Projects Agency*). C'est pour cette raison que vous le trouverez aussi sous l'appellation *DoD Model* pour *Department of Defense Model* (« modèle du département de la Défense »). Quant au modèle OSI, il a été créé en 1978 par l'Organisation internationale pour la standardisation (ou ISO, *International Organization for Standardization*). C'est un certain Charles Bachman qui proposa la segmentation de la communication dans un réseau en sept couches distinctes.

## II. Un modèle qui en tient une couche

Les buts de ces deux modèles ne sont pas les mêmes. En effet, le modèle OSI a été développé à vocation normative, c'est-à-dire pour servir de référence dans le déroulement de la communication entre deux hôtes. D'ailleurs, il est également connu sous les noms *OSI Reference model* (« modèle de référence OSI ») ou OSI-RM. Alors que le modèle TCP/IP a une vocation descriptive, c'est-à-dire qu'il décrit la façon dont se passe la communication entre deux hôtes. En d'autres termes, si vous voulez comprendre comment se déroule la communication « sur le terrain », prenez le modèle TCP/IP. Par contre, si vous voulez comprendre la suite logique, la procédure selon la norme, penchez-vous sur le modèle OSI. Ceci dit, c'est le modèle OSI qui vous servira de « plan » si vous voulez créer un protocole ou un matériel en réseau.

i

Il se peut qu'*Internet Reference Model* fasse parfois référence au modèle TCP/IP. Cette appellation n'est pas fautive, mais inexacte : la suite protocolaire TCP/IP sert de description plutôt que de référence.

### II.2.4.2. Comparaison dans la structure

Voici un schéma comparatif des deux modèles.

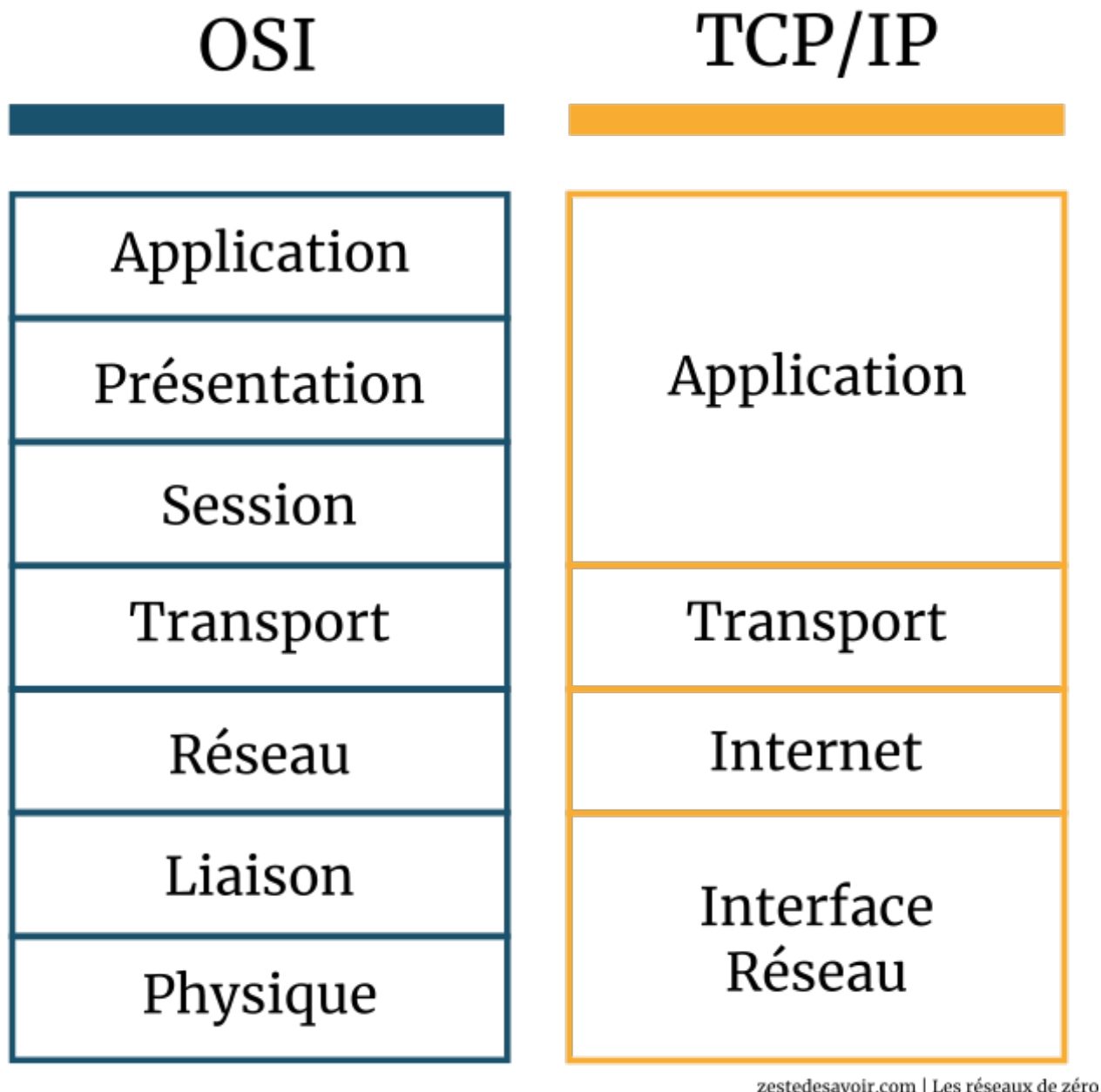


FIGURE II.2.4. – Comparaison OSI vs TCP/IP (CC BY)

Comme vous le voyez, le modèle TCP/IP n'est constitué que de quatre couches. Ce sont des couches d'abstraction, autrement dit des couches qui cachent les détails d'implémentation de la communication et leurs noms ne reflètent pas mot pour mot les fonctions qu'elles assurent. Le modèle OSI, quant à lui, est fièrement constitué de sept couches. Les trois premières couches du modèle OSI correspondent à la couche applicative du modèle TCP/IP.

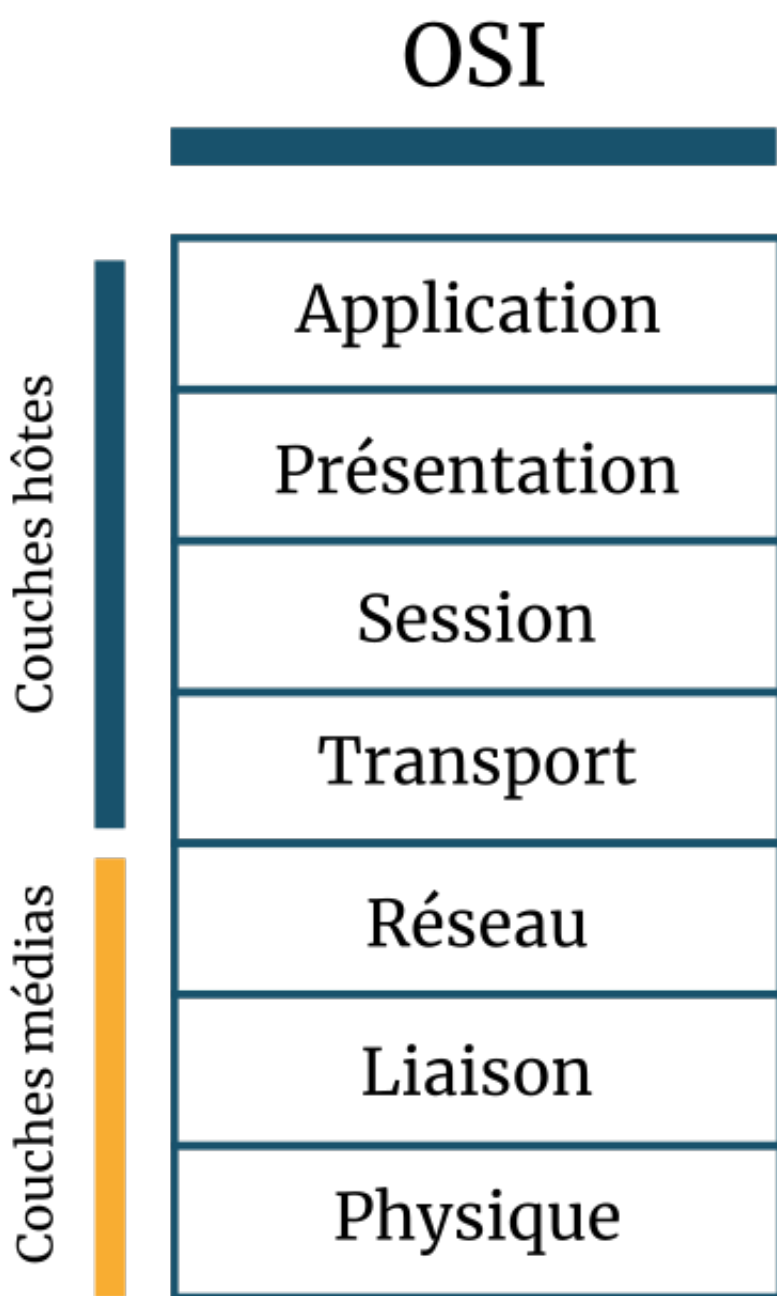


Cette correspondance ne veut pas dire que la couche applicative du modèle TCP/IP soit une synthèse des trois premières couches du modèle OSI. Non ! Elle ne remplit que les rôles des couches application et présentation du modèle OSI, comme le spécifie la [RFC 1122](#) [↗](#).



Le formatage des données dans le modèle TCP/IP peut également se faire *via* des bibliothèques.

Les deux modèles possèdent une couche de transport. La couche réseau du modèle OSI correspond à la couche Internet(work) du modèle TCP/IP. Les couches liaison de données et physique du modèle OSI forment une seule couche pour le modèle TCP/IP : interface réseau. Les couches application, présentation, session et transport sont dites « couches hôtes » (*host layers* en anglais). En effet, ces couches « concernent » directement les hôtes. Les couches réseau, liaison et physique, elles, sont des couches de médias (*media layers*) : elles sont plus liées au média qu'à l'hôte. Voici un schéma illustrant cette correspondance :



## II. Un modèle qui en tient une couche

FIGURE II.2.5. – Différenciation entre couches médias et couches hôtes sur le modèle OSI (CC BY)

### II.2.4.3. Point vocabulaire: les unités de données

Au début de la communication entre deux hôtes, chaque information qui sera transmise est une donnée. Cependant, cette donnée a plusieurs unités selon la couche dans laquelle elle se trouve : il s'agit de la même donnée, mais sous plusieurs appellations. Prenons un exemple : votre père, vous l'appellez papa à la maison. Au travail, on l'appelle M. X ; chez son frère, ses neveux l'appellent tonton, etc. C'est bien la même personne, connue sous plusieurs appellations selon le milieu.

Ainsi, les données que vous transmettez sont tout simplement appelées unité de données (*data unit* en anglais). On les nomme parfois PDU (*Protocol Data Unit* : « unité de données de protocole ») ; dans ce cas, leur nom sera précédé de l'initiale de la couche dont ces données sont issues. Par exemple dans la couche applicative, elles prennent le nom d'APDU (*Application Protocol Data Unit* : « unité de données de protocole d'application »). Dans la couche de session, elles s'appelleront donc... SPDU (*Session Protocol Data Unit* : « unité de données de protocole de session »). Même principe pour la couche de présentation. Une fois dans la couche de transport, où elles sont segmentées, ces données deviennent logiquement des segments. (Nous les avons appelés séquences dans le chapitre précédent.)



L'appellation TPDU (*Transport Protocol Data Unit*) est également correcte en ce qui concerne la couche de transport.

Dans la couche réseau du modèle OSI, ces données prennent le nom de paquets ; dans les couches liaison et physique, respectivement ceux de *frame* (trame) et *bit*.

Voici une image résumant cela ~~pour votre plus grand plaisir~~. 🍊 Les acronymes dans l'image ci-dessous sont en anglais parce qu'ils sont plus courts. 🍌 Vous ne devriez pas avoir de difficulté à les comprendre puisque leurs équivalents français sont juste plus haut.

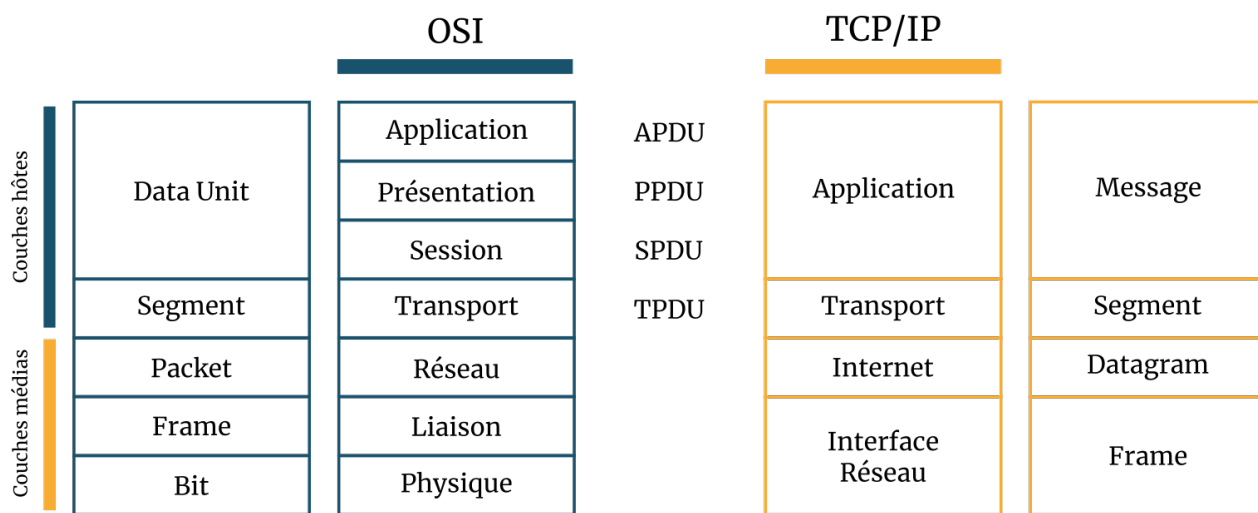


FIGURE II.2.6. – Unités de données selon les modèles OSI et TCP/IP (CC BY)

## II. Un modèle qui en tient une couche

i

Vous pouvez remarquer la présence de *datagram* dans le schéma. *Datagram* (datagramme) est le nom donné à un PDU transmis par un service non fiable (UDP par exemple). On pourra, dans certains cas, retrouver ce terme sur la couche transport. Mais le moment n'est pas encore venu. 🍊

i

Tout au long du tutoriel, nous utiliserons souvent les mots « données » et « paquets » pour faire référence à toute information qui se transmet. Vous vous rendrez vite compte qu'il est difficile de faire autrement. L'utilisation du mot approprié interviendra lorsqu'elle sera de rigueur.

### II.2.4.4. Faites attention à l'abstraction des noms de couches

Les noms des couches des modèles TCP/IP ou OSI sont abstraits, voilà pourquoi nous vous avons parlé de couches d'abstraction. Leurs noms ne sont pas toujours synonymes de leurs fonctions et peuvent par moments être vagues. Par exemple, la couche application du modèle OSI ne veut pas dire grand-chose. Quand vous lisez *application*, est-ce que cela vous donne une idée de la fonction de cette couche ? Ce nom n'est pas si explicite. La couche transport des deux modèles est certainement la plus abstraite dans sa dénomination. Quand on lit *transport*, on a tendance à croire que cette couche transporte vraiment les données jusqu'à son destinataire — alors que la transmission s'effectue à la couche 1 (physique) du modèle OSI et à la couche interface réseau du modèle TCP/IP. Par contre, la couche réseau est la moins abstraite, l'on comprend tout de suite qu'il s'agit de l'exercice des fonctions intimement liées au réseau.

### II.2.4.5. Critiques du modèle OSI

En dehors de l'abstraction des noms de couches, dont le modèle TCP/IP est également coupable, les reproches faits à ce modèle relèvent de quatre domaines : la technologie, l'implémentation, la durée de recherche et l'investissement.

#### II.2.4.5.1. La technologie

Par technologie, nous voulons parler de degré de complexité. Le modèle OSI est plus complexe que le modèle TCP/IP. En effet, sept couches contre quatre : y a pas photo ! 🍊 Cette complexité peut faire douter de l'utilité de certaines couches. Par exemple, les couches présentation et session sont assez rarement utilisées. Lorsque l'ISO a voulu « neutraliser » la normalisation/standardisation du modèle OSI, les Britanniques n'ont pas hésité à demander la suppression de ces couches-là. Comme nous l'avons vu en survolant les couches de ce modèle, certaines fonctions se partagent entre plusieurs niveaux. Par conséquent, la complexité même du modèle OSI réduit l'efficacité de la communication.

## II. Un modèle qui en tient une couche

### II.2.4.5.2. L'implémentation

À cause de la complexité de ce modèle, ses premières implémentations ont été très difficiles, lourdes et surtout lentes.

### II.2.4.5.3. La durée et l'investissement

En technologie, il faut sortir le bon produit au bon moment, n'est-ce pas ? OSI n'a pas respecté cette règle. Les recherches de l'ISO pour mettre au point un modèle normalisé ont pris du temps : OSI est sorti alors que le modèle TCP/IP était déjà utilisé. De ce fait, l'ISO a rencontré des difficultés pour trouver un investissement, le monde n'étant pas tellement intéressé par une deuxième suite de protocoles.

### II.2.4.6. Critiques du modèle TCP/IP

N'allez pas croire que le modèle TCP/IP est parfait ! Nous pouvons lui reprocher plusieurs choses :

- Contrairement au modèle OSI, TCP/IP ne distingue pas clairement le concept de services réseaux, des interfaces et des protocoles. Par conséquent, il ne respecte même pas la bonne procédure de l'ingénierie logicielle.
- Le modèle TCP/IP est un peu « carré ». Nous voulons dire par là qu'il est tellement spécifique que l'on ne peut pas se servir de ce modèle pour en décrire un autre, alors que le modèle OSI peut être utilisé pour décrire le principe du modèle TCP/IP.
- Interface réseau : c'est ainsi que l'académie Cisco appelle cette couche du modèle TCP/IP. La [RFC 1122](#) la nomme tout simplement lien ; on la trouve aussi sous l'appellation hôte-à-réseau (*host-to-network*). Cette couche a été fortement critiquée parce qu'il ne s'agit pas d'une couche à proprement parler, mais d'une interface entre le réseau et la liaison de données.
- Le modèle TCP/IP ne fait pas la distinction entre la couche physique et la couche liaison de données. En principe, la couche physique devrait être une couche à part, car elle « conclut » la transmission grâce à la mise sur média.

### II.2.4.7. Et maintenant: le verdict des juges

Après avoir comparé les deux modèles, l'heure est à ~~la sanction~~ au verdict !

En conclusion à cette analyse/critique des deux modèles, il est clair que TCP/IP a plus de succès qu'OSI. Mais ce succès est simplement dû au fait que les protocoles de ce modèle sont les plus utilisés. Sans ses protocoles, le modèle TCP/IP serait pratiquement inexistant. Par contre, le modèle OSI, avec ou sans protocoles, est la parfaite norme dictant la procédure de communication. Plusieurs personnes ont sanctionné le modèle OSI au profit de TCP/IP et, d'après elles, TCP/IP gagnerait ce duel.

*Le paragraphe suivant a été écrit par Jean Pouabou, auteur historique du cours*

## II. Un modèle qui en tient une couche

Cependant, je ne partage pas cet avis, et après quelques recherches fructueuses, je me déclare pro-OSI. Je voterais même pour le remplacement du modèle TCP/IP. La seule chose que je peux reprocher au modèle OSI, qui est encore d'actualité, est la présence des couches présentation et session — qui sont presque inutiles. Sans elles, le modèle OSI serait, pour moi, le modèle idéal. Cette conviction est également fondée sur le rapport analytique publié en 2004 par Internet Mark 2 Project, intitulé *Internet Analysis Report 2004 - Protocols and Governance* (« Rapport de l'analyse d'Internet - Protocoles et gouvernance »). Vous pouvez télécharger un résumé de ce rapport gratuitement [ici](#) et le rapport complet (en anglais) se trouve [là](#).



L'analyse en soi est très critiquable. À votre niveau, vous ne serez peut-être pas capable d'en proposer une autre. Ce n'est pas grave. Cependant, notez qu'il y a matière à réflexion dans certaines remarques.



Si le modèle OSI est meilleur que le TCP/IP, pourquoi ce dernier a-t-il plus de succès ?

TCP/IP est sorti, et fut donc largement utilisé, avant le modèle OSI. De cette utilisation massive découle une complexité de migration vers un autre modèle, d'où le maintien du succès de TCP/IP.



Je ne comprends pas l'anglais, mais je veux lire le rapport de l'analyse. Une solution ?

Oui : apprendre l'anglais ! 🍌

### II.2.5. Principe d'encapsulation

Chaque couche du modèle OSI a une fonction déterminée. Nous avons vu que la couche en cours utilise les services de la couche au-dessous d'elle qui, à son tour, en offre pour la couche du dessous. Cette corrélation indique bien que certaines informations peuvent se retrouver d'une couche à une autre. Cela n'est possible que grâce au principe d'encapsulation.

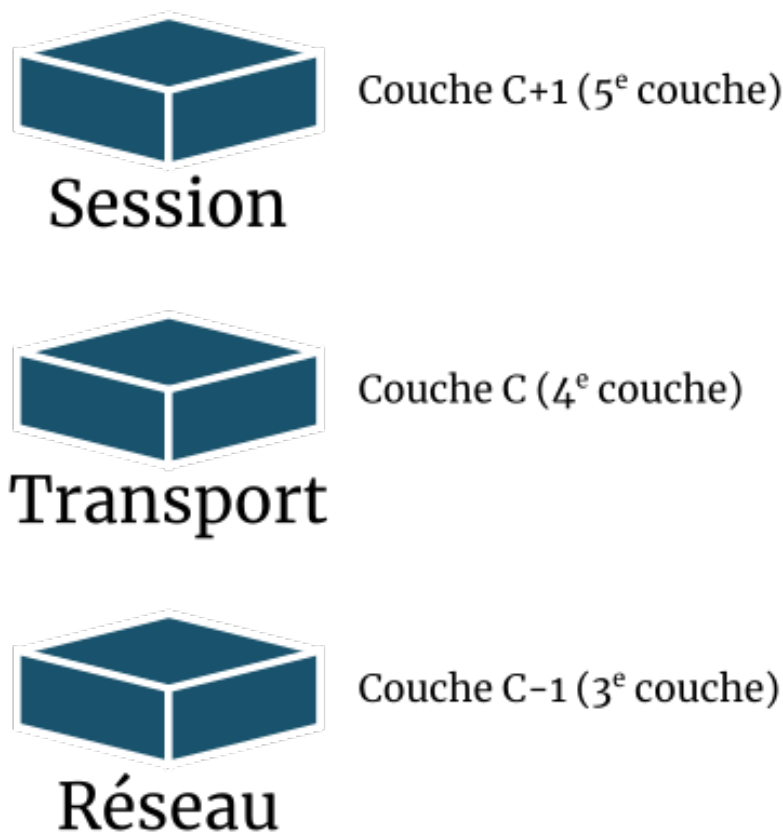
L'encapsulation consiste à encapsuler. 🍌 En d'autres termes, elle consiste à envelopper les données à chaque couche du modèle OSI.

Quand vous écrivez une lettre (pas un mail), vous devez la glisser dans une enveloppe. C'est à peu près le même principe dans le modèle OSI : les données sont enveloppées à chaque couche et le nom de l'unité de données n'est rien d'autre que le nom de l'enveloppe. Nous avons vu dans la sous-partie précédente que, dans la couche applicative, l'unité de données était l'APDU (ou plus simplement le PDU). Ensuite, nous avons vu que dans la couche réseau, l'unité de données était le paquet. Ces PDU forment une sorte d'enveloppe qui contient deux choses : la donnée en elle-même et l'en-tête spécifique à cette couche. La partie « donnée » de ce paquet est composée de la donnée initiale, mais aussi des en-têtes des couches qui la précèdent. Il existe une toute

## II. Un modèle qui en tient une couche

petite formule mathématique définissant la relation entre les couches. Ce n'est pas difficile, pas la peine de fuir !

Considérons l'image ci-dessous :



zestedesavoir.com | Les réseaux de zéro

FIGURE II.2.7. – Relation entre couches (CC BY)

Soit  $C$  une couche du modèle OSI. La couche  $C + 1$  utilise les services de la couche  $C$ . Facile, n'est-ce pas ? La couche session utilise les services de la couche transport, par exemple. La donnée que la couche  $C + 1$  transmet à la couche  $C$  est appelée **SDU** tant qu'elle n'a pas encore été encapsulée par cette dernière. Si, par contre, la couche  $C$  encapsule ce **SDU**, on l'appelle désormais... PDU.

?

Quelle est donc la relation entre le PDU et le **SDU** ?

Dans une couche  $C$ , le PDU est le **SDU** de la couche  $C + 1$  plus son en-tête (couche  $C$ ). Ce **SDU** ne devient un PDU qu'après l'encapsulation. La couche  $C$  ajoute des informations dans l'en-tête (*header*) ou le pied (*trailer*), voire les deux, du **SDU** afin de le transformer en un PDU. Ce PDU sera alors le **SDU** de la couche  $C - 1$ . Donc le PDU est un **SDU** encapsulé avec un en-tête.

Voici la constitution d'un PDU :

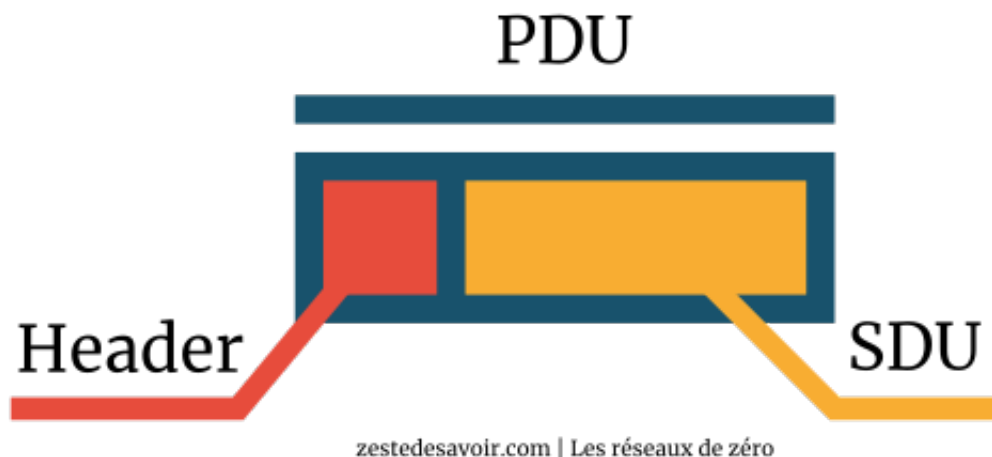


FIGURE II.2.8. – Constitution d'un PDU (CC BY)

Comprendre la relation entre un **SDU** et un PDU peut être complexe. Pour vous simplifier la tâche, nous allons considérer un exemple inspiré du monde réel et vous aurez ensuite droit à un schéma.



Nous classons l'exemple ci-dessous entre les catégories « un peu difficile » et « difficile ». Il est important de ne pas admirer les mouches qui voltigent dans votre bureau en ce moment. Soyez concentrés. 🍌

Quand vous écrivez une (vraie) lettre, c'est un **SDU**. Vous la mettez dans une enveloppe sur laquelle est écrite une adresse. Cette lettre qui n'était qu'un **SDU** devient un PDU du fait qu'elle a été enveloppée (encapsulée). Votre lettre arrive à la poste. Un agent du service postal regarde le code postal du destinataire et place la lettre dans un sac. Mais on ne la voit plus, puisqu'elle est dans un sac. Pour l'instant, la lettre, l'enveloppe et le sac forment un **SDU**. L'agent du service postal va alors inscrire le code postal du destinataire sur le sac en question, qui devient donc un PDU. S'il contient d'autres lettres partant pour la même ville, elles seront alors toutes mises dans une caisse : c'est un **SDU**. Tout comme on a ajouté des informations sur l'enveloppe et sur le sac, il faut également mettre un code postal sur la caisse. Cet ajout fait de cette caisse un PDU.

Voilà pour la procédure de transmission. Mais pour la réception, les sacs à l'intérieur de la caisse (des **SDU**) sont enlevés lorsqu'elle atteint sa destination. **Attention, c'est ici que vous devez être très attentif • ve.** Si un individu prend un sac et en lit le code postal pour l'acheminer à son destinataire, le sac **n'est plus** considéré comme un **SDU** mais comme un PDU. C'était un **SDU** au moment de sa sortie de la caisse. Étant donné qu'il y a des informations de plus sur le sac, c'est un PDU pour celui qui les lit.

Lorsque le destinataire recevra la lettre, les informations ajoutées sur le sac ou sur la caisse ne seront plus visibles : il ne restera plus qu'une enveloppe contenant la lettre originale (un **SDU**).

Tenez, un schéma illustrant l'encapsulation des **SDU** dans le modèle OSI :

## II. Un modèle qui en tient une couche

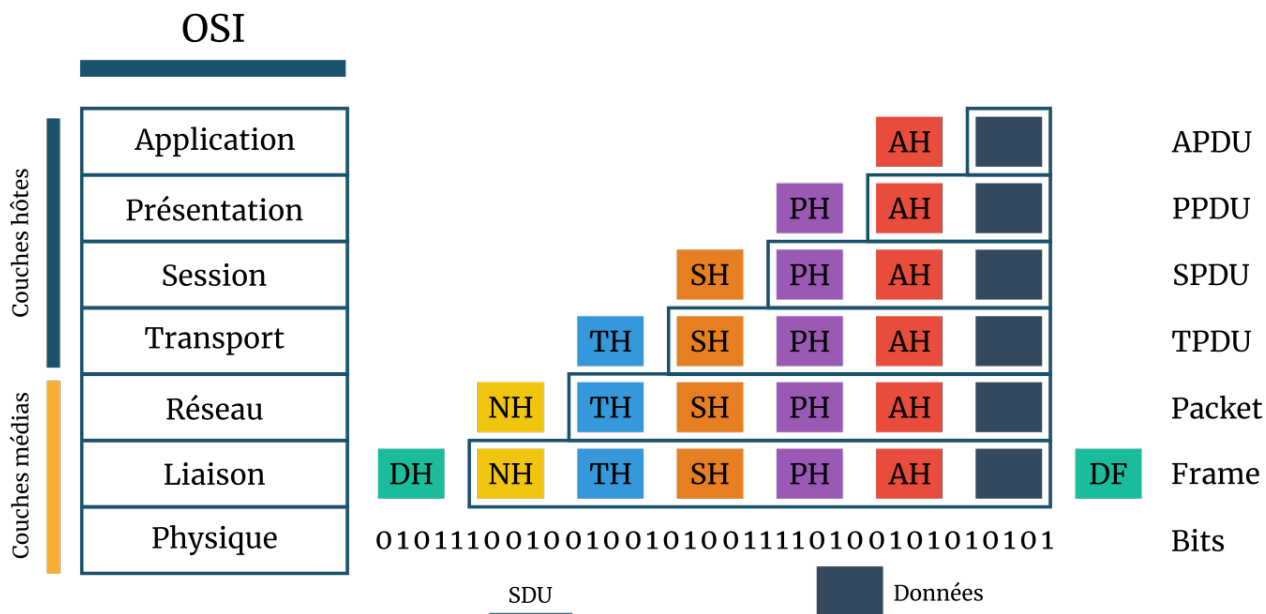


FIGURE II.2.9. – Encapsulation des SDU (CC BY)



Dans le schéma ci-dessus, DF signifie *Data link Footer*. Le terme n'est pas exact, mais nous l'utilisons pour faciliter votre compréhension. Le vrai terme français qui équivaut au mot *trailer* est « remorque ». Une remorque est un genre de véhicule que l'on attèle à un autre véhicule ; la remorque est en quelque sorte la queue ou le *footer* du véhicule principal. Il est donc plus facile d'utiliser *footer* plutôt que *trailer*, le mot pied plutôt que remorque.



Tous les éléments encadrés en bleu forment un **SDU**, comme le stipule la légende.

Comme vous le voyez, au début nous n'avons que les données initiales, que l'on pourrait également appeler données d'application. La donnée initiale à ce stade est un **SDU**. Une fois dans la couche applicative, un en-tête AH (*Application Header* : « en-tête d'application ») est ajouté à cette donnée initiale. La donnée de la couche applicative est un APDU. La couche applicative transmet cela à la couche de présentation au-dessous. Cette donnée transmise est un **SDU**. Par l'encapsulation, cette couche ajoute un en-tête PH au **SDU** de la couche applicative. La couche de présentation envoie ce « nouveau » message à la couche de session et cette dernière encapsule son *header* avec le **SDU** obtenu de la couche présentation pour former son SPDU. Et ainsi de suite jusqu'à la couche liaison, qui a la particularité d'ajouter également un *trailer*. Finalement, toutes ces données sont converties en une série de *bits* et mises sur le média pour la transmission.



Une couche ne doit pas connaître (ne connaît pas) l'existence de l'en-tête ajouté par la couche au-dessus d'elle (la couche C + 1). En fait, cet en-tête, par l'encapsulation, apparaît comme faisant partie intégrante de la donnée initiale. Par conséquent, la couche

## II. Un modèle qui en tient une couche

i

ignore qu'il s'agit d'un en-tête, mais elle le considère comme appartenant aux données à transmettre.

Vous pouvez également constater que toutes les informations ajoutées dans la couche supérieure se retrouvent dans la couche inférieure. Ainsi dans la couche réseau, par exemple, on retrouve la donnée initiale + l'en-tête d'application (AH) + PH + SH + TH. Toutes ces « informations » seront considérées par la couche réseau comme la donnée initiale. Dans cet exemple, la couche réseau ne s'occupe donc que de son propre en-tête.

?

Si, à chaque couche, l'en-tête est ajouté à la donnée initiale, ne serait-ce pas compromettre l'intégralité du message ?

Qui peut répondre à cela ? 🍊 Très belle question, soit dit en passant. 🍊 Chaque couche ajoute à la donnée initiale un en-tête. De la sorte, tous les en-têtes sont réunis dans la couche de liaison. Lorsque ces informations seront converties en une suite de *bits*, le récepteur devrait recevoir des données erronées puisque la donnée initiale n'avait pas tous ces en-têtes, n'est-ce pas ? En principe. Mais le modèle OSI (ou le modèle TCP/IP) est assez intelligent. En effet, dans la procédure de réception, chaque en-tête est enlevé lorsque le message « grimpe » les couches, tel qu'illustré par le schéma ci-dessous. Cette « suppression » d'en-tête, c'est la décapsulation !

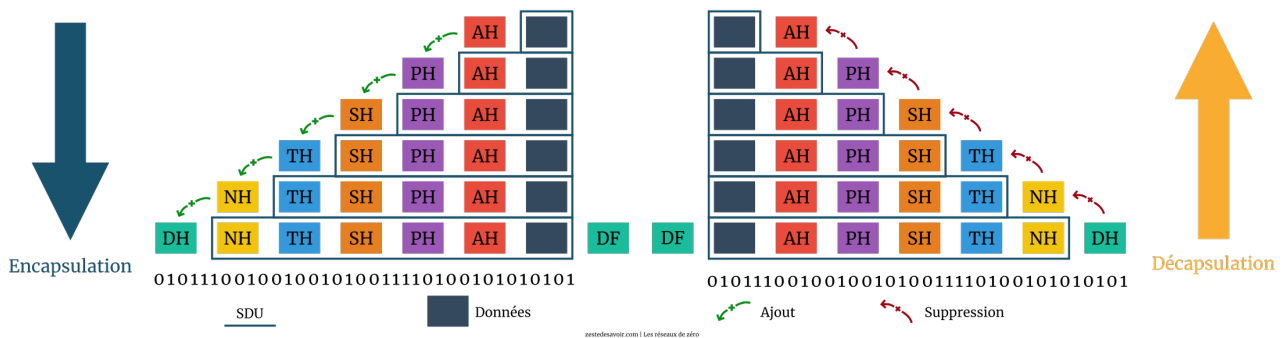


FIGURE II.2.10. – Représentation schématique de l'encapsulation et de la décapsulation (CC BY)

Comme vous le voyez sur le schéma, dans la procédure de réception, chaque couche supprime son en-tête correspondant **après l'avoir lu**. Par exemple, l'en-tête NH (réseau) est supprimé dans la couche réseau de l'hôte récepteur après que ce dernier l'a lu.

## Conclusion

Maintenant que vous savez à quoi il sert, nous allons entrer dans les coulisses du modèle OSI par le haut. Pourquoi pas par le bas ? Parce qu'il est plus facile de descendre des escaliers que de les monter. Parce que nous estimons qu'il est plus intéressant de commencer par ce qui est plus proche de nous, à savoir les applications que nous utilisons.

## II.3. De l'application à la session

### Introduction

Dans ce chapitre, nous allons étudier les trois dernières couches du modèle OSI, à savoir, de haut en bas : la couche applicative (7), la couche de présentation (6) et la couche de session (5).

#### II.3.1. Rôle des couches

Les couches 7, 6 et 5 du modèle OSI correspondent à une seule couche applicative dans le modèle TCP/IP, voilà pourquoi nous allons les étudier dans une même sous-partie.

##### II.3.1.1. Couche 7 : application

À votre grande surprise, vous apprendrez que cette couche n'a pas de rôle défini. En fait, il s'agit seulement d'une couche-interface. Le terme « interface », comme nous l'avons vu, sous-entend que cette couche sert de point de contact ou d'interaction entre l'utilisateur que vous êtes et les services en réseaux. Par exemple, votre navigateur web est une application qui vous permet d'être en contact avec un service offert par le protocole HTTP (*HyperText Transfer Protocol*). Quand vous utilisez votre messagerie, vous êtes en interaction avec la couche applicative.

La couche applicative héberge principalement :

- Des API (*Application Programming Interface* : « interface de programmation d'application ») : une API est grosso-modo un ensemble de fonctions permettant à un programme externe d'interagir avec un programme interne pour ne pas exposer le code source. Vous n'êtes pas obligés de savoir ce qu'est une API, mais notez que les API offrant des fonctions de réseaux se trouvent dans la couche application.
- Des services : plusieurs services en réseaux tels que la navigation Internet, la messagerie, la résolution de noms de domaine sont concentrés dans cette couche.

Cette couche regorge de protocoles tels que :

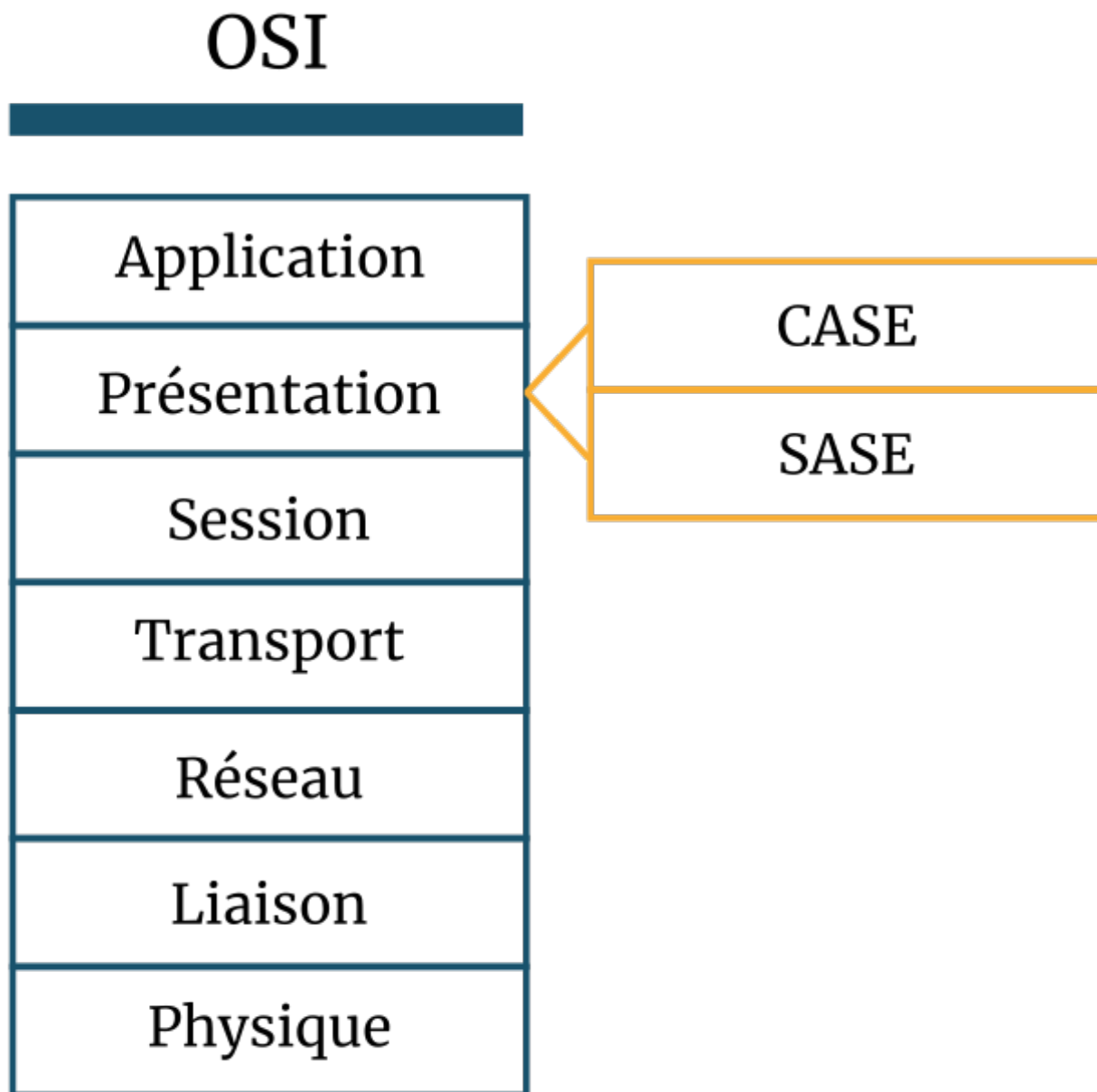
- FTP ;
- HTTP ;
- TFTP ;
- Telnet ;
- SMTP.

Nous n'allons pas étudier tous les protocoles de cette couche (il y en a tellement...) mais en examiner quelques-uns qui sont plutôt intéressants et simples à comprendre.

### II.3.1.2. Couche 6 : présentation

Le nom de cette couche est suffisamment explicite : la couche 6 s'occupe de tout ce qui a trait à la présentation. En d'autres termes, elle offre des services permettant de convertir des données d'un système d'encodage à un autre (de l'EBCDIC vers l'ASCII, par exemple), de compresser des fichiers, de les crypter, etc. Lorsque vous utilisez Winzip, un logiciel de compression, vous utilisez un service de la couche 6 du modèle OSI. Par conséquent, c'est dans cette couche que nous trouvons des protocoles — que nous n'allons pas étudier — tels que LPP (*Lightweight Presentation Protocol*), NDR (*Network Data Representation*) ou encore NCP (*NetWare Core Protocol*).

Un détail souvent omis lorsqu'on traite cette couche est qu'elle se subdivise en deux (sous-)couches. Détail peu important puisque l'union de ces deux sous-couches forme la couche en elle-même. Cependant, afin d'enrichir vos connaissances, voici un schéma illustrant les deux couches qui composent la couche présentation du modèle OSI.



zestedesavoir.com | Les réseaux de zéro

FIGURE II.3.1. – Couche 6 du modèle OSI (CC BY)

La sous-couche CASE (*Common Application Service Element* : « élément de service pour les applications courantes ») se charge d'offrir des services pour... les applications courantes ; tandis que SASE (*Specific Application Service Element* : « élément de service pour une application spécifique »), comme son acronyme l'indique, offre des services pour des applications précises. Si l'explication vous paraît ambiguë, ne vous en faites pas : vous comprendrez tout de même la suite du tutoriel. 🍏

### II.3.1.3. Couche 5 : le gestionnaire de session

Tout est dans le titre. La couche 5 du modèle OSI a la responsabilité de l'ouverture, de la fermeture et de la gestion des sessions entre les applications. Les deux services principaux offerts

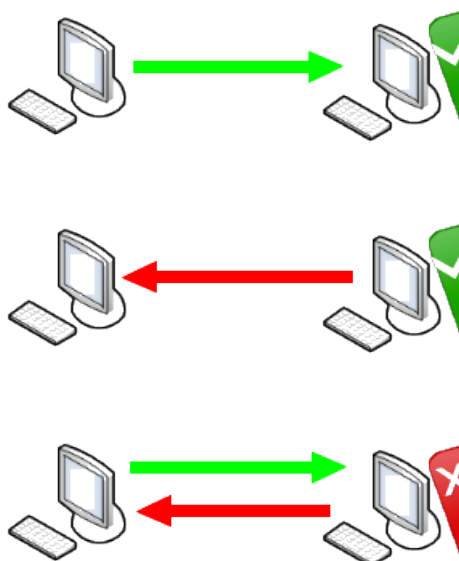
## II. Un modèle qui en tient une couche

par cette couche sont la gestion des sessions (ouverture, fermeture, restauration) et la gestion des permissions (quelle application peut faire quoi).

Les protocoles de la couche 5, tels que X.225, peuvent déterminer la direction de la communication. Il existe deux types de communication :

- *Half Duplex* (HDX) : système de communication permettant l'échange par tour. Si deux entités A et B sont membres d'un réseau fondé sur ce système de communication, elles ne peuvent pas échanger de données au même moment. Chacune doit attendre son tour !
- *Full Duplex* (FDX) : l'exact contraire du HDX. A et B peuvent communiquer simultanément sans que cela ne pose problème.

Voici deux schémas illustrant ces deux systèmes de communication.



Les réseaux de zéro - zestedesavoir.com

FIGURE II.3.2. – Schéma illustrant le half-duplex



Les réseaux de zéro - zestedesavoir.com

FIGURE II.3.3. – Schéma illustrant le full-duplex


*i*

Sachez qu'il existe un autre système de communication appelé **simplex**. Nous avons préféré le mettre de côté puisqu'il n'est pas utile pour la suite du tutoriel.

## II.3.2. BitTorrent, le protocole de partage

Vous savez maintenant à quoi sert la couche applicative des modèles TCP/IP et OSI. Mais nous ne vous avons pas inculqué le concept de cette couche pour nous arrêter en si bon chemin. 🍊 Nous allons donc explorer quelques protocoles de cette couche, en commençant par un protocole de partage : le célèbre BitTorrent.

### II.3.2.1. La naissance de BitTorrent

Conçu par [Bram Cohen](#) , BitTorrent est un protocole permettant le partage de fichiers de taille importante. BitTorrent est sans conteste le protocole de partage le plus utilisé sur Internet et ne vous est certainement pas inconnu. Créé en 2001, son développement continu est assuré par l'entreprise BitTorrent, Inc. Avec BitTorrent, l'échange ou le partage de fichiers se fait dans une infrastructure *Peer2Peer* (« pair-à-pair »). Par opposition à une architecture centralisée, le pair-à-pair relie les hôtes directement en formant une sorte de topologie maillée.

?

Pourquoi avoir créé BitTorrent ?

Le succès de BitTorrent est sans doute dû à la minimisation de surcharge du réseau de partage. Imaginez un serveur qui hébergerait 10 000 vidéos. Que se passerait-il si un million d'utilisateurs téléchargeraient simultanément la même vidéo sur ce serveur ? Il aurait à répondre à un million de requêtes à la fois, ce qui ralentirait significativement le réseau de partage. Plus le nombre d'internautes essayant d'accéder à un même fichier au même moment est grand, plus le fichier devient difficilement accessible à cause de la congestion du réseau. C'est de ce constat qu'est né le protocole BitTorrent.

### II.3.2.2. Le fonctionnement de BitTorrent

Et si chaque utilisateur devenait à la fois client et serveur ? Telle est la question que le créateur de BitTorrent a dû se poser. Le fonctionnement de ce protocole de partage est en effet le suivant : si un utilisateur X télécharge un film Y provenant d'un serveur Z, les autres utilisateurs pourront télécharger le même film à travers X pour ne pas alourdir le serveur de base Z.

Pour mieux comprendre ce principe, voici une animation illustrant un réseau utilisant un protocole de partage classique (client-serveur) :

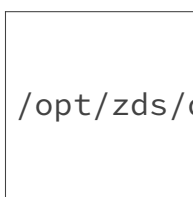


FIGURE II.3.4. – Téléchargement simple

## II. Un modèle qui en tient une couche

Comme vous pouvez le voir dans cette animation, le serveur envoie quatre copies de ladite vidéo aux quatre clients.

À présent, voici une animation décrivant un partage *via* BitTorrent.

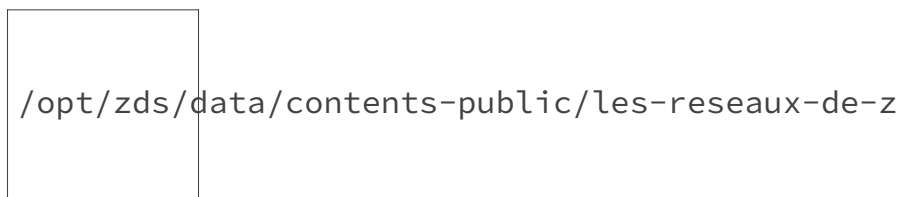


FIGURE II.3.5. – Animation illustrant un partage BitTorrent

BitTorrent minimise la congestion du réseau en coupant le fichier en plusieurs portions. Tous les clients en reçoivent une, puis ils font office de serveurs les uns pour les autres jusqu'à ce que chaque client ait reçu toutes les portions du fichier. Certes, les portions seront reçues dans le désordre, mais BitTorrent est assez intelligent pour les réagencer correctement. C'est ce qu'on appelle le contrôle de séquence (ça vous dit quelque chose ?). 🍏 BitTorrent est donc un protocole très pratique et économe. Pas étonnant que Facebook et Twitter l'utilisent pour la distribution des mises à jour sur leurs serveurs !

### II.3.2.3. La terminologie de BitTorrent

En matière de réseaux, le vocabulaire est très important. Nous allons donc parcourir quelques termes propres au protocole que nous étudions.



Les termes de BitTorrent sont vraiment interdépendants : chacun d'eux peut faire référence à un autre, qui en appelle un autre à son tour, etc. Soyez concentrés pour ne pas vous embrouiller.

#### II.3.2.3.1. Les semences et les semeurs

Vous avez certainement déjà rencontré les termes *seed* et *seeder*. *Seed* est un mot anglais signifiant « semence ». Un *seeder* est un pair (en anglais *peer*) dans le réseau de partage qui a en sa possession une copie complète d'un fichier. Le *seeder* a la totalité du fichier en partage, alors que le *peer* n'a en général qu'une partie dudit fichier. Dans notre animation, chaque ordinateur qui détient une portion de la vidéo est un *peer*. À la fin du téléchargement, il devient *seeder* étant donné qu'il a acquis la totalité de la vidéo. Le *seeder* est donc un « semeur » qui distribue un *seed* dans le réseau, comme un jardinier répartirait des semences à la surface de la terre.

## II. Un modèle qui en tient une couche

### II.3.2.3.2. Les essaims

Avez-vous déjà entendu l'expression « essaim d'abeilles » ? Un essaim est un groupement important d'insectes d'une même famille. Par exemple, les zAgrumes sont un essaim : ils forment un groupement important d'insectes sur un même site. 🍊 Avec BitTorrent, un essaim (*swarm* en anglais) est formé par les *peers* partageant un même torrent. Si sept *seeders* et sept autres *peers* ont tous un torrent en commun, ils forment un essaim de quatorze unités.

### II.3.2.3.3. Le traqueur: Big Brother

Un *tracker* (« traqueur ») n'est rien d'autre qu'un serveur dans le réseau de partage. Cependant, il n'est pas directement impliqué dans la transmission ou dans le partage — d'ailleurs, il ne possède pas de copie du fichier partagé. En quelque sorte, il sert de policier en gardant en mémoire les traces laissées par les *seeds* et les *peers* de l'essaim. Il informe également les clients, desquels il reçoit des comptes rendus périodiques, de la présence d'autres clients auxquels ils peuvent se connecter.

### II.3.2.3.4. Les sangsues et les lâches




Il y a des sangsues dans un protocole ? 🍊

Oh que oui ! Une sangsue (*leech* en anglais) est un ver qui se nourrit du sang d'autres êtres vivants. Dans un réseau de partage, on qualifie de sangsue tout client qui télécharge plus qu'il ne partage. On parle également de *lurker* (de l'anglais *to lurk* : « se cacher », « se dissimuler »). En gros, c'est un lâche. On utilise le terme *lurker* pour faire référence à un client qui télécharge sans ajouter de contenu nouveau au réseau. La différence entre un *lurker* et un *leech(er)* est assez mince. Un *leech* décide parfois de lui-même de ne plus semer après avoir téléchargé, alors que le *lurker*, même s'il n'uploadé aucune nouveauté, a la bonne pratique de partager ce qu'il télécharge.

### II.3.2.3.5. Le ratio de partage (share ratio)

Le ratio de partage permet d'évaluer la contribution d'un client à un réseau de partage. Il est obtenu en divisant le nombre de partages par le nombre de téléchargements. Il est souhaitable qu'un client ait un ratio de partage supérieur à 1, c'est-à-dire qu'il partage plus qu'il ne télécharge. En revanche, un ratio de partage inférieur à 1 veut dire qu'un client est plus impliqué dans le téléchargement que dans le partage.

En ce qui concerne les termes propres à BitTorrent, nous allons nous arrêter là. Nous vous invitons à faire une recherche sur Internet à chaque fois que vous tomberez sur l'un des termes que nous n'avons pas abordés. Pour ceux qui désirent poursuivre l'exploration de BitTorrent, nous vous recommandons [le tutoriel de Natim](#)  (lien d'archive) qui est bien plus complet que notre présentation. 🍊

### II.3.3. SMTP : le protocole de transmission de mail

Le service de messagerie (instantanée ou non) est sans doute le plus utilisé de nos jours quotidiennement. Enfin, en dehors de Facebook. 🍊 Chacun de nous est amené à consulter ses mails régulièrement, voire à en rédiger. La messagerie électronique nous a facilité la tâche en réduisant le temps de rédaction et d'acheminement d'un courrier. Ça vous dirait de voir comment ça se passe dans les coulisses ? Nous allons donc étudier le fonctionnement d'un protocole nous permettant d'envoyer un message électronique.

#### II.3.3.1. Présentation rapide de SMTP

SMTP (*Simple Mail Transfer Protocol* : « protocole simple de transfert de courrier ») a été créé dans les années 1970, aux débuts d'Internet. Comme tout bon protocole qui se veut être un standard, il a fallu qu'il soit spécifié par une requête de commentaires (RFC). C'est donc en 1982 qu'il est spécifié par la [RFC 821](#) 📄. Une RFC de 2008 comprenait des mises à jour de la première implémentation de ce protocole : la [RFC 5321](#) 📄.

SMTP a commencé à être massivement utilisé au début des années 1980. Il sert principalement à envoyer des mails. Comme son nom l'indique, il s'agit d'un protocole de transmission et non de réception. Cependant, les serveurs de messagerie utilisent SMTP pour faire les deux, c'est-à-dire la transmission **et** la réception, cette dernière n'étant en fait qu'une transmission, n'est-ce pas ? Comment ça, vous ne le saviez pas ? 🍊 En voici une démonstration :

- Vous écrivez une lettre (physique) à un ami au Japon.
- La lettre arrive à la poste centrale du Japon.
- La poste centrale va à son tour transmettre la lettre au domicile de votre camarade.

Vous avez confié la lettre à la poste, qui l'a envoyée au destinataire. Pour vous et pour la poste, il s'agit d'une transmission ; pour votre ami, d'une réception. La réception n'est donc qu'une autre transmission. 🍊

Ainsi, les serveurs de messagerie utilisent SMTP pour la transmission et la réception, tandis que les clients de messagerie utilisent SMTP pour l'envoi et un autre protocole (POP ou IMAP) pour la réception. Nous allons étudier ces protocoles de retrait dans la sous-partie suivante.

?

SMTP sert donc à transmettre un mail, mais n'a-t-il pas besoin d'utiliser un protocole de transmission ?

La probabilité que vous vous soyez posé cette question est proche de zéro. 🍊 Mais bon... SMTP est un protocole de transfert. Or pour transférer, il faut un autre protocole de transmission.

?

Un protocole peut en utiliser un autre ? 🦉

Les protocoles qui assurent la transmission se trouvent dans la couche de transport. Par conséquent, un protocole de transfert de la couche application (comme SMTP) ne peut se passer d'un protocole de transmission de la couche transport (UDP ou TCP). Nous étudierons et comparerons justement les protocoles de transport dans la partie 4 du cours.



Si vous avez décroché à partir des protocoles de transmission, ce n'est pas grave : nous verrons cela de manière plus claire dans le prochain chapitre. Mais il faut raccrocher au protocole SMTP maintenant !

### II.3.3.2. Cheminement d'un courriel

Maintenant, nous allons voir les étapes par lesquelles passe un courriel avant d'atteindre son destinataire. Comme il est de coutume dans ce tutoriel, nous allons commencer par une analogie qui ne doit pas vous être inconnue. Pierre (encore lui) habite à Paris. Il veut écrire une lettre à André, qui habite Lyon. Dans notre scénario, la procédure de transmission/réception de la lettre se fera ainsi :

- Pierre écrit la lettre.
- Le facteur vient la chercher.
- La lettre arrive à la poste locale.
- La poste envoie la lettre à Lyon.
- Elle arrive à la poste locale de Lyon.
- Un facteur est chargé de la porter au domicile d'André.
- André reçoit la lettre.

Hormis l'étape 1 (écriture de la lettre) et l'étape 7 (réception), la lettre est passée entre les mains du facteur, de la poste locale, de la poste distante (celle de Lyon) et d'un autre facteur. Soit quatre étapes.



Vous pouvez constater qu'il y a deux facteurs dans notre exemple : le premier peut être qualifié de « facteur de transmission », car il est impliqué dans l'étape de transmission de notre lettre ; le second, de « facteur de réception », car il est impliqué dans la transmission/réception de notre lettre.

Voici un schéma illustrant ces étapes :

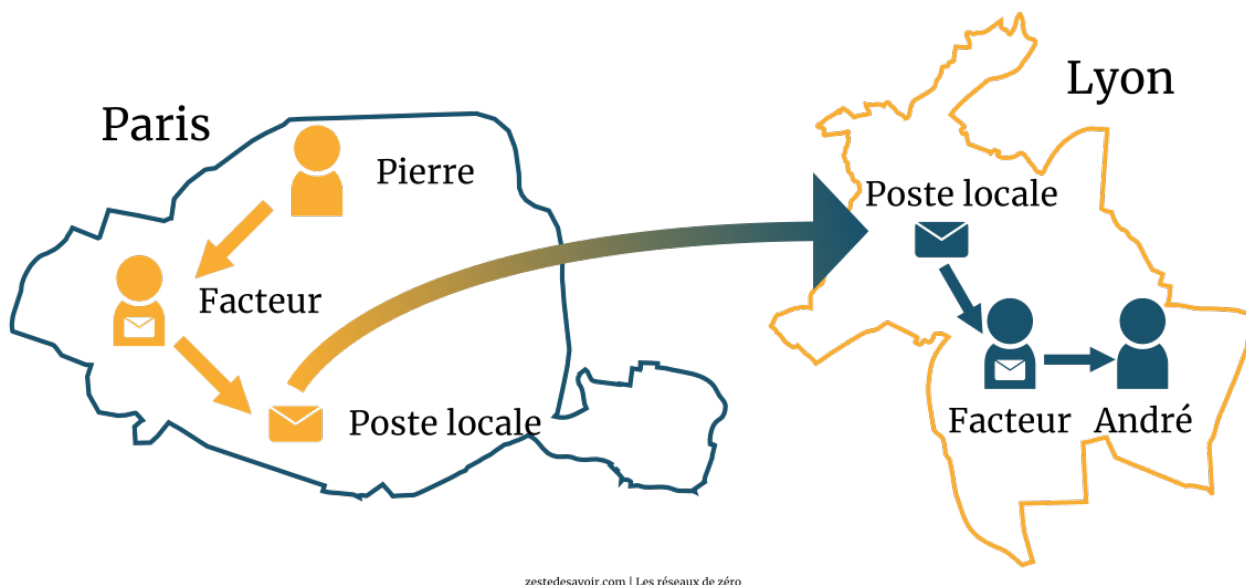


FIGURE II.3.6. – Étapes dans le cheminement d'un courrier (CC BY)

?

Et alors, où est le rapport avec le protocole SMTP ?

Avez-vous déjà oublié que la technologie s'inspire de la vie quotidienne ? Le protocole SMTP suivra exactement le même principe et votre mail passera lui aussi par quatre étapes : le facteur, la poste locale, la poste distante et un autre facteur (voir schéma suivant).

?

Quoi, il y a des bureaux de poste dans les réseaux ? 🍊

Mais bien sûr. C'est d'ailleurs pour cela qu'il existe un protocole POP, acronyme de *Post Office Protocol*, soit littéralement « protocole de bureau de poste ».

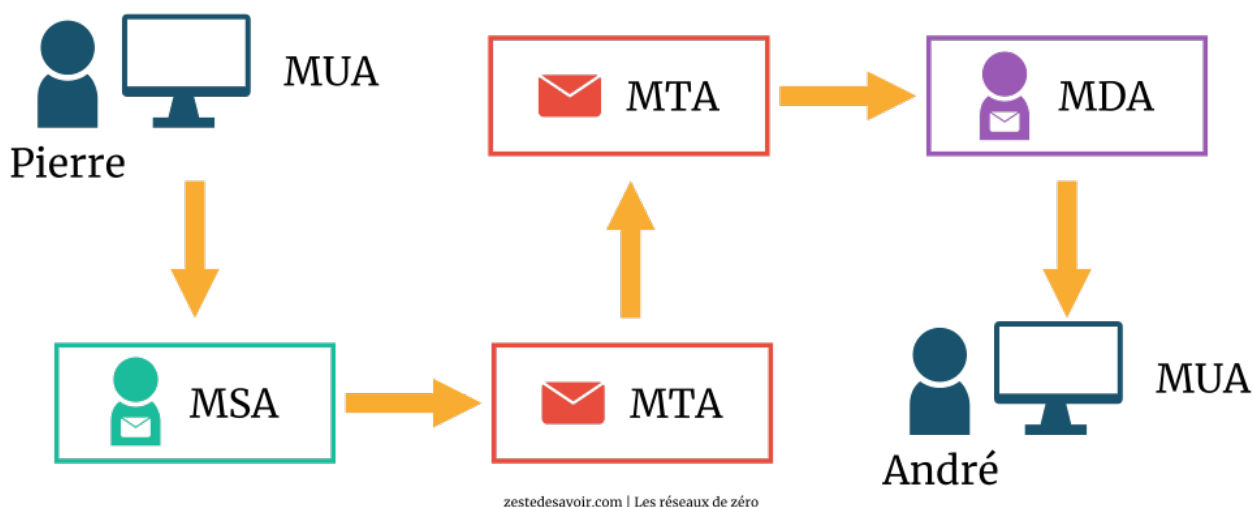


FIGURE II.3.7. – Positionnement des MTA, MUA, MSA et MDA (CC BY)

## II. Un modèle qui en tient une couche

?

D'abord vous nous dites que notre courriel passera par le facteur, la poste locale, etc. Mais maintenant vous nous parlez de MUA, de MTA... 🍊

Et alors ? Remplacez X par sa valeur ! 🍊

Le facteur de transmission correspond sur le schéma au MSA, les deux bureaux de poste sont des MTA et le facteur de réception est un MDA.

?

Ok, voulez-vous bien nous expliquer tout ça ? Et qu'est-ce que le MUA ?

Pas de panique !

### II.3.3.3. Commençons par le MUA

Considérons que le papier et le stylo que vous utilisez pour écrire une lettre forment une « application de rédaction ». Dans la couche applicative, qui nous sert d'interface avec les services réseaux, le MUA (*Mail User Agent* : « client de messagerie ») n'est autre que l'application de rédaction d'un courriel, un client de messagerie. C'est une application, comme Outlook ou Thunderbird, qui vous permet de retirer des mails de votre boîte de réception et d'en écrire.

?

Je n'utilise aucun logiciel pour le retrait de mes mails, je le fais directement sur Hotmail/Yahoo/Gmail/autre. Alors, qu'en est-il ?

Hotmail & cie sont également des MUA, plus précisément des *webmails*. Ce sont des applications auxquelles on accède par l'intermédiaire d'un navigateur.

Alors, dans notre schéma, Pierre utilise Outlook/Yahoo/autre pour écrire un courriel et clique sur... « Envoyer ». Direction : le MSA !

### II.3.3.4. C'est quoi, ce MSA ?

MSA signifie *Mail Submission Agent*, soit « agent de soumission de courrier ». Comme son nom l'indique, son rôle est donc de soumettre le mail. O.K., mais à qui ? Dans notre scénario, le MSA est le facteur de transmission. Votre lettre ne peut pas quitter directement Pékin pour la terre Adélie, n'est-ce pas ? Surtout qu'il n'y a aucun habitant en terre Adélie. C'est pour l'exemple. 🍊 Le facteur se chargera de la conduire à la poste où les mesures nécessaires d'envoi seront prises pour l'acheminement de votre courrier à son destinataire, dans les plus brefs délais et dans des conditions normales de température et de pression satisfaisantes. Un MSA n'est en fait qu'un autre logiciel-interface, c'est-à-dire un intermédiaire entre votre client de messagerie et le serveur de messagerie (serveur Gmail ou autre).

Il est possible de fusionner un MTA et un MSA. Dans ce cas, on parle seulement de MTA — mais ce dernier assure également le rôle d'un MSA. Comme si, pour en revenir à notre scénario,

## II. Un modèle qui en tient une couche

vous décidiez d'aller vous-même déposer votre lettre à la poste locale au lieu de passer par un facteur.

### II.3.3.5. MTA, l'agent de Jack Bauer de transfert

Un MTA ou *Mail Transfer Agent* est l'agent de transmission du courriel, tout comme le bureau de poste est l'agent de transmission de votre courrier. Mais le bureau de poste reçoit également les courriers externes pour les expédier à leurs destinataires respectifs. Voilà pourquoi nous voyons sur le schéma que le courrier passe d'un MTA à un autre, de la même manière que la lettre de Pierre passe du bureau de poste local de Paris à celui de Lyon.

Quand vous écrivez un mail à une personne dont l'adresse appartient à un autre domaine que la vôtre, il passe par un second MTA. Cependant, lorsqu'il s'agit d'un mail interne à un même domaine, il est directement pris en charge par le MDA sans passer par le second MTA. Un exemple ? Si Pierre écrit un courrier à Jacques et qu'ils habitent tous deux à Paris, la lettre ira à la poste locale *via* le facteur de transmission. Une fois à la poste, on enverra simplement un facteur de réception livrer la lettre puisqu'il est inutile de passer par la poste d'une autre ville.

Voici un schéma illustrant le transfert d'un mail entre deux clients dans un même domaine.

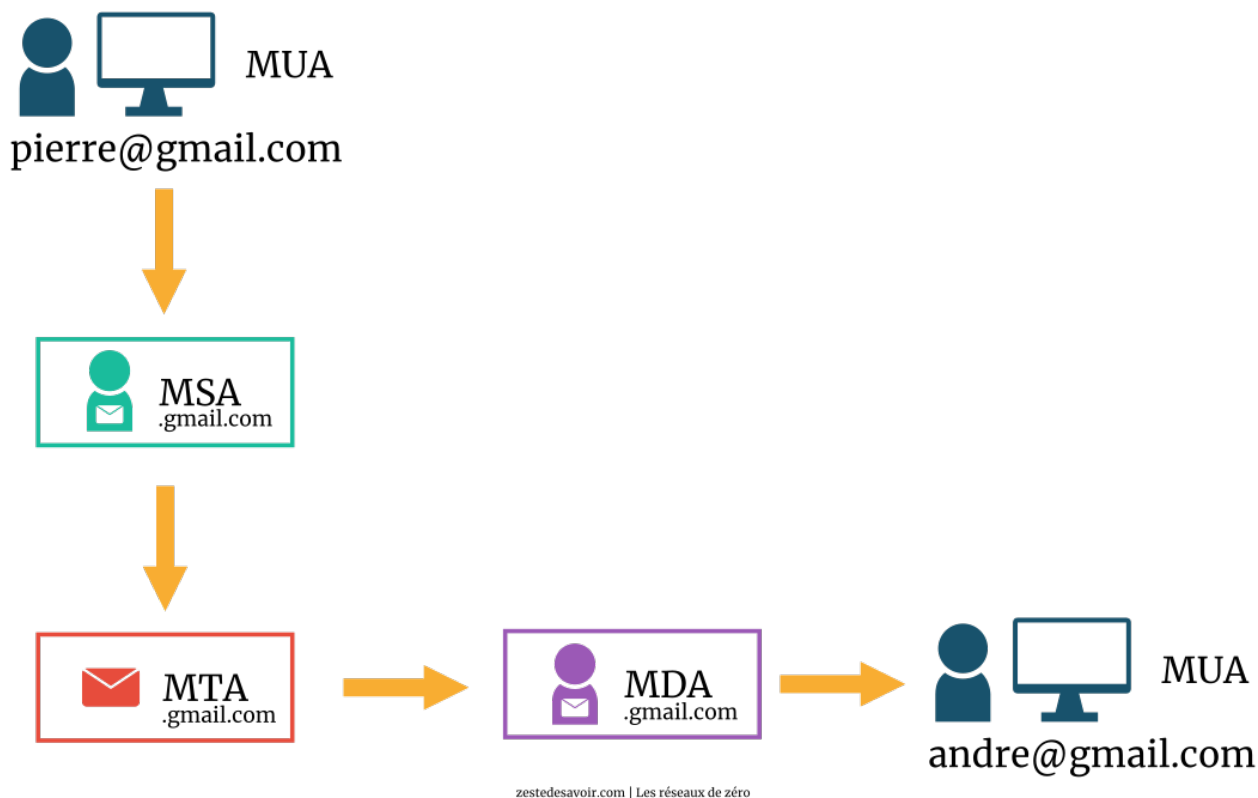


FIGURE II.3.8. – Transmission d'un e-mail au sein d'un même domaine (CC BY)

Voici à présent un autre schéma illustrant le transfert d'un mail entre deux clients de domaines différents.

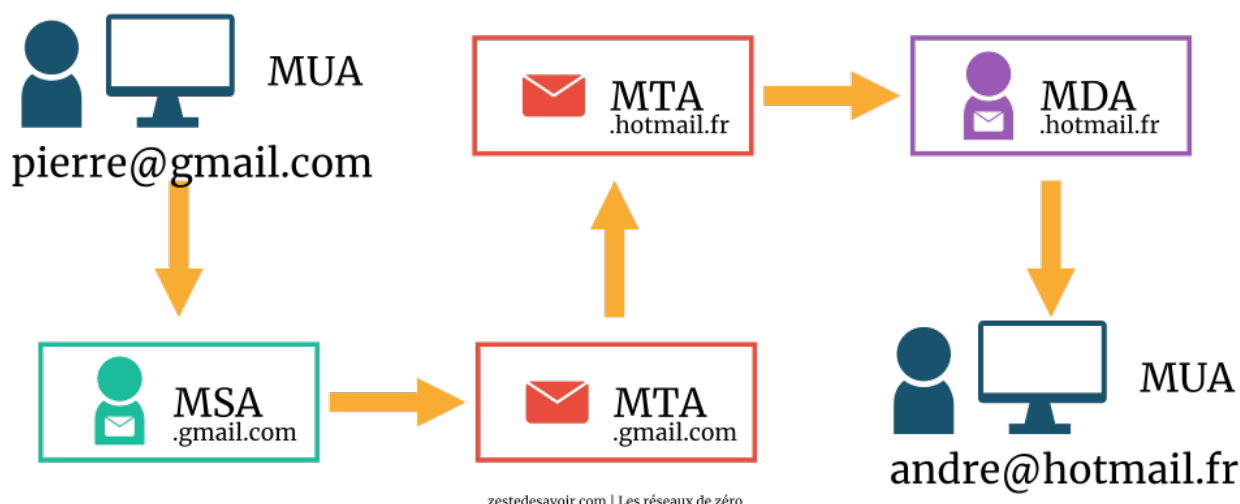


FIGURE II.3.9. – Transmission d'un e-mail entre deux domaines différents (CC BY)

Le MTA de Gmail étudiera la partie qui se trouve après le caractère @ dans l'adresse du destinataire afin de vérifier s'il s'agit d'un transfert de mail à un client du même domaine (un client Gmail en l'occurrence). Il se rendra compte que hotmail.fr ne concerne pas son domaine et enverra donc le courriel au MTA du domaine de Hotmail.

i

Nous reverrons les noms de domaine dans une autre partie de ce cours. En attendant, vous pouvez lire le tutoriel très complet de Mathieu Nebra sur la gestion du nom de domaine [↗](#).

### II.3.3.6. Pour terminer: le grand MDA

Pour comprendre ce qu'est le MDA (*Mail Delivery Agent* : « agent livreur de courrier »), posons-nous deux questions.

?

Qu'est-ce qu'un livreur de pizzas ?

Un livreur de pizzas, c'est quelqu'un qui livre des pizzas. 🍕

?

Quelle est la différence entre le facteur de transmission et le facteur de réception dans notre schéma ?

Voilà une question sérieuse ! Les deux sont des facteurs, les deux livrent donc des pizzas du courrier. Sauf que vous envoyez vous-même le premier (facteur de transmission) livrer le courrier, alors que le second (facteur de réception) est envoyé par le bureau de poste. Vous commandez une pizza, la demande est traitée et on vous envoie un livreur, n'est-ce pas ? Mais ce n'est pas vous qui avez donné l'ordre directement au livreur de vous apporter la pizza.

## II. Un modèle qui en tient une couche

Voilà pourquoi nous faisons une nette distinction entre les deux facteurs. L'un s'appelle MSA et l'autre MDA pour les raisons évoquées ci-dessus. Nous disons au MSA : « Écoute, va transmettre ce mail à X » ; tandis que le MTA dit au MDA : « Tiens, X a reçu un mail, viens le chercher et le stocker ». Tout est clair ?

i

Nous pourrions aussi considérer le MDA comme une vraie boîte aux lettres où les courriers sont stockés en attendant que leur destinataire vienne les chercher. Cependant, en vertu du scénario établi, le MDA sera considéré comme le facteur de réception qui vient déposer le courrier dans votre boîte aux lettres.

### II.3.3.7. Quand les protocoles s'emmêlent...

Du MUA de Pierre au dernier MTA impliqué dans le processus de transmission, c'est le protocole SMTP qui est utilisé. Entre le MDA et le dernier MUA (celui d'André), c'est un **protocole de réception** qui est utilisé : POP, POP2 ou IMAP.

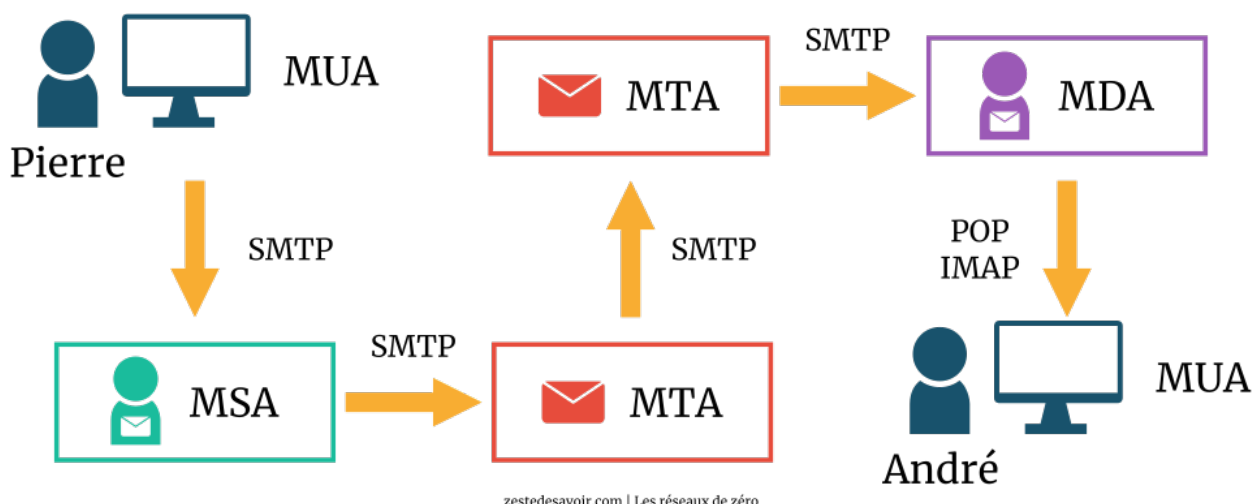


FIGURE II.3.10. – Protocoles utilisés dans la transmission d'un e-mail (CC BY)

Vous pouvez également vous rendre compte que les MTA utilisent SMTP pour la transmission **et** la réception comme nous l'avons indiqué un peu plus haut. Suivez-nous dans la prochaine sous-partie pour une exploration de ces protocoles de retrait de mail !

### II.3.4. IMAP vs POP : les protocoles de retrait de mail

Nous allons vous présenter seulement les protocoles POP et IMAP dans une même sous-partie, car ils servent à faire la même chose et que chacun présente des avantages que l'autre n'a pas. Commençons par le protocole POP.

### II.3.4.1. Le bureau de poste version électronique : présentation

POP (*Post Office Protocol* : « protocole de bureau de poste ») a l'avantage d'être simple et efficace et surtout, il est supporté par tous les clients de messagerie. Pour comprendre le rôle assuré par ce protocole, nous allons examiner le rôle d'un bureau de poste dans la vie courante.

?

Quels sont les services offerts par un bureau de poste ?

La question peut paraître idiote, mais quand on sait qu'en France, La Poste est aussi une banque et un opérateur de téléphonie mobile... 🍊 Cependant, nous n'allons retenir que les services qui concernent directement notre étude. Le bureau de poste a pour fonction principale de traiter les courriers : il les reçoit et les distribue à leurs destinataires respectifs. Il est également en contact avec les autres bureaux de poste distants.

En réseau, en ce qui concerne la messagerie électronique, le protocole POP fait plus ou moins la même chose. La différence avec un véritable service postal est que deux bureaux de poste de villes différentes peuvent échanger du courrier, alors que le protocole POP ne peut pas en envoyer. POP n'est qu'un protocole de retrait : il permet d'aller chercher un mail se situant sur un serveur de messagerie, mais pas d'en envoyer. L'envoi est assuré par le protocole SMTP.

Il existe trois versions de ce protocole : POP1, POP2 et POP3. La toute première version est spécifiée par la [RFC 918](#) . POP2 est spécifié par la [RFC 937](#) et vous pouvez retrouver les spécifications de POP3 dans la [RFC 1081](#) . La spécification actuelle de ce protocole se trouve dans la [RFC 1939](#) , qui n'est en fait que la RFC 1081 incluant un mécanisme de gestion d'extension et un autre d'authentification. La [RFC 2595](#) sécurise le protocole POP en l'utilisant de pair avec le protocole SSL (*Secure Socket Layer*) : on parle donc aussi de POP3S, avec un S pour SSL. Le protocole SSL a depuis évolué en TLS (*Transport Layer Security*). Nous n'allons pas étudier en détail le protocole POP : il existe déjà un [tutoriel à ce sujet](#) que nous vous invitons à lire.

Le protocole POP permet bien sûr de récupérer son courrier, mais aussi d'en laisser une copie sur le serveur. Cela est particulièrement utile si l'on ne peut plus accéder pour une raison quelconque (panne...) aux e-mails déjà téléchargés : on peut toujours les télécharger de nouveau. Néanmoins, il n'a pas vraiment été conçu pour cela, contrairement à IMAP.

### II.3.4.2. IMAP : un protocole qui a la tête dans les nuages

IMAP (*Internet Message Access Protocol*) est un protocole de lecture d'e-mails. Contrairement à POP, il n'a pas été conçu pour recevoir des messages mais pour les consulter directement depuis un serveur. Cette consultation s'apparente à du *cloud*, c'est-à-dire l'accès par Internet à des données qui ne se trouvent pas sur notre disque dur. IMAP est assez avancé puisqu'il permet de gérer ses messages directement sur un serveur distant pour organiser nos messages en dossiers, par exemple. Il supporte également TLS. Dans le cas d'IMAP, le *clouding* est à la fois un avantage et un inconvénient : on peut accéder à ses messages depuis n'importe quel ordinateur, à condition d'être connecté à son compte de messagerie. Quelques clients permettent néanmoins de télécharger les messages pour pallier ce problème. Certains clients de messagerie ne gèrent pas très bien le protocole IMAP, qui est défini par la [RFC 3501](#) . Vous pouvez avoir plus de détails sur ce protocole en lisant [ce tutoriel](#) .

## Conclusion

Voilà qui est fait ! On descend encore une marche ? Ah, on nous demande de nous identifier pour accéder à la couche suivante... Mais comment faire ? Eh bien, suivez-nous dans la prochaine partie pour découvrir l'identification et l'adressage, puis nous pourrons accéder à la couche 4 !

# Conclusion

Faisons une pause dans notre descente des couches du modèle OSI. Nous allons maintenant étudier l'identification et l'adressage, qui sont des notions clés pour pouvoir descendre plus bas !

## Troisième partie

Veillez vous identifier pour communiquer

# Introduction

Nous arrivons à un moment où l'identification et l'adressage deviennent des éléments clés pour pouvoir aller plus loin. C'est pourquoi nous allons y consacrer toute une partie !

## III.1. Des adresses en folie !

### Introduction

Pour communiquer, il faut savoir à qui on veut s'adresser ! Lorsque nous avons parlé du commutateur (ou *switch*) dans le chapitre sur le matériel, nous avons évoqué des moyens d'identification au sein du réseau : l'adresse IP et l'adresse MAC. Il est temps de voir ce que c'est, et pourquoi on a besoin de ces 2 types d'adresses.

#### III.1.1. IP vs MAC

Il est temps de parler de l'identification et de la communication dans un réseau. Nous allons aborder 2 notions : il s'agit des adresses IP et des adresses MAC. Nous allons les aborder une par une, et comprendre pourquoi il y a des adresses IP et des adresses MAC.

##### III.1.1.1. Adresse IP : l'adresse relative au réseau

Dans le premier chapitre, nous avons vu un exemple simple de la transmission d'un livre entre humains. Mais, pour transmettre un livre à André, vous devez savoir où il habite.

Une adresse IP n'est « rien d'autre » que l'endroit où habite un ordinateur. Mais attention : cette adresse est *relative au réseau*. Une machine n'aura pas forcément la même adresse IP sur un réseau X et un réseau Y. Nous n'entrerons pas dans les détails pour le moment, l'adressage n'étant pas vraiment une base.

Les adresses IP sont le seul moyen d'identification des machines sur Internet. Mais il existe 2 versions du protocole Internet (la « manière » d'accéder à Internet en quelque sorte) : IPv4 et IPv6. Et chaque version utilise sa propre structure d'adresse IP.

Une « adresse IPv4 » est constituée de 4 nombres correspondant à **4 octets** compris entre 0 et 255, séparés par des points. Exemple : 88.45.124.201. De nos jours, ce sont les plus connues. Les « adresses IPv6 » sont encore plus complexes : elles sont représentées par une suite de 8 groupes de 2 octets représentés en hexadécimal (je vous avais prévenu que c'était complexe 🍊). Exemple (tiré de Wikipédia) : 1fff:0000:0a88:85a3:0000:0000:ac1f:8001.

Cette explication de ce qu'est une adresse IP est acceptable pour l'instant, mais vous verrez pourquoi une adresse IP est plus complexe que ça. En fait elle agit un peu comme un signe distinctif : si dans une rue toutes les maisons sont identiques, comment faites-vous pour reconnaître celle d'André ou de Pierre ? Dans notre exemple, c'est en se basant sur le numéro affiché devant la maison. Mais s'il existe plusieurs rues ? Plusieurs maisons peuvent avoir le même numéro

### III. Veuillez vous identifier pour communiquer

sans être au même emplacement. Comment fait-on pour délimiter les rues ? On utilise pour cela un **masque de sous-réseau**, et l'adresse IP correspond au numéro de chacune des maisons.

i

Internet est une sorte de rue géante, comportant des croisements avec d'autres rues plus petites. Ces petites rues sont des sous-réseaux connectés à Internet, et chaque messenger (chaque passerelle par défaut) aux carrefours possède une adresse IP spéciale relative au réseau Internet.

#### III.1.1.2. Adresses MAC : l'adresse relative à la carte réseau

!

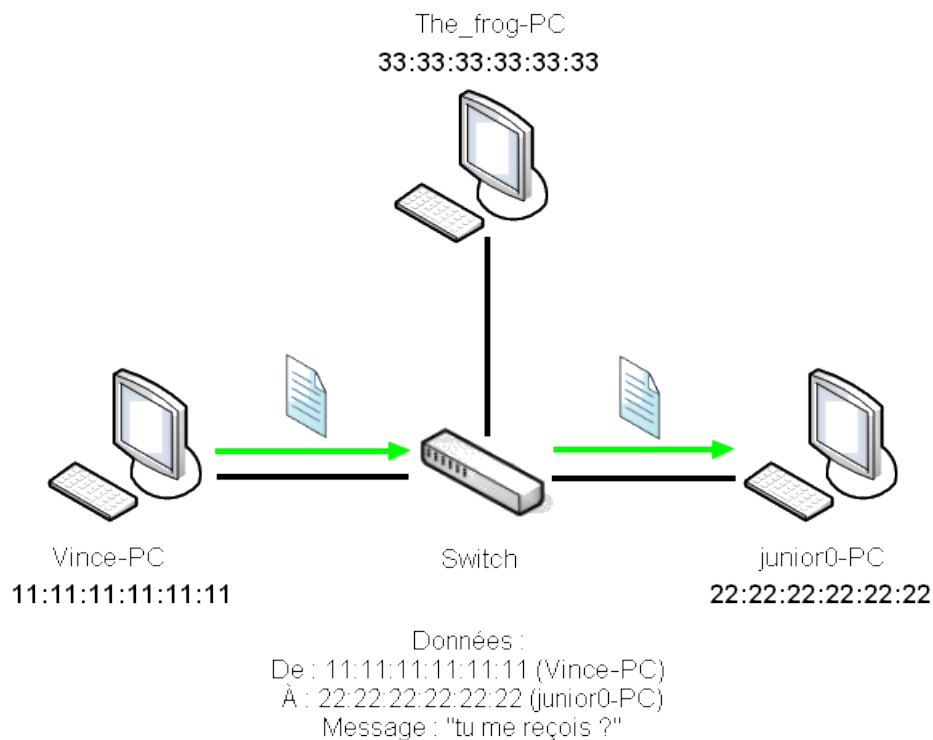
Précisons avant tout, le nom d'adresse MAC n'a rien à voir avec les ordinateurs Mac. Il vaut mieux prévenir, on ne sait jamais... 🍊

Comme dit brièvement lors du chapitre précédent, une adresse MAC est un identifiant unique attribué à chaque carte réseau. C'est une adresse **physique**. Concrètement, c'est un numéro d'identification composé de 12 chiffres hexadécimaux. Par convention, on place un symbole deux-points ( : ) tous les 2 chiffres. Une adresse MAC ressemble donc à cela : **01:23:45:67:89:AB**.

Imaginons un petit réseau de 3 ordinateurs connectés au même *switch*. Rappelez-vous, un *switch* est plus intelligent qu'un *hub*. Plutôt que d'envoyer ce qu'il reçoit par un port à tous les autres, il « filtre » les données renvoyées en se basant sur les adresses MAC des ordinateurs qui sont connectés. Prenons par exemple trois ordinateurs. Appelons-les Vince-PC, junior0-PC, et The\_frog-PC (au cas où vous vous demanderiez pourquoi ces noms, ce sont les auteurs historiques du tuto 🍊 ). Si Vince-PC veut communiquer avec junior0-PC, il va envoyer au *switch* ce qu'il veut communiquer à junior0-PC. Le *switch*, ou commutateur, va regarder l'adresse MAC du destinataire et va lui envoyer ce qui lui est destiné sans l'envoyer aux autres machines (ici à The\_frog-pc). En fait, le commutateur utilise une table de correspondance entre adresses MAC et ports pour savoir où envoyer les données.

Voici une illustration d'une communication basée sur les adresses MAC :

### III. Veuillez vous identifier pour communiquer



Les réseaux de zéro - zestedesavoir.com

FIGURE III.1.1. – Une trame contient au moins les adresses MAC du destinataire et de l'expéditeur

?

Mais pourquoi on n'utilise pas juste les adresses MAC ?

Parce que dans un grand réseau, comme un WAN, ou même Internet, il n'y a pas d'élément central qui connaît l'emplacement du destinataire et qui peut renvoyer les données en conséquence. Par contre, le système d'adresses IP permet, grâce à un processus appelé **routing**, d'assurer que les données arrivent bien au destinataire. Le routage sera expliqué dès la prochaine partie.

#### III.1.1.3. En résumé...

La différence primordiale entre les adresses IP et les adresses MAC est que les adresses IP sont routables. Elles peuvent communiquer avec des machines au-delà d'un sous-réseau, contrairement aux adresses MAC. Le *switch*, au cœur du LAN, se base donc sur les adresses MAC pour assurer la communication entre plusieurs machines appartenant à un même sous-réseau. En revanche, les adresses IP sont utilisées pour faire communiquer des machines de sous-réseaux différents.

On espère que vous avez compris. 🍊

#### III.1.2. Masque de sous-réseau et passerelle

Afin de présenter ces notions, nous allons reprendre l'idée d'un « réseau » d'humains.

### III.1.2.1. Les sous-réseaux et leurs masques

Considérons deux personnes, Jacques et Jean, et un gros réseau : leur ville. Nous allons établir des lois. Pour que deux personnes puissent se parler directement :

- Elles doivent parler la même langue ;
- Elles doivent habiter dans la même rue ;
- Chaque personne doit connaître l'adresse de l'autre (le numéro de la maison de l'autre).

Si Jacques habite la rue ClemStreet, et Jean aussi, alors ils peuvent facilement communiquer : ce n'est pas bien loin, ils vont marcher mais ils doivent, évidemment, parler la même langue. La rue est ici l'équivalent de ce qu'on appelle en informatique un **sous-réseau**, quant à la langue, c'est ce que l'on appelle un **protocole**. Si vous avez bien compris : une autre rue, par exemple juniorStreet (c'est le créateur du tuto qui a choisi ce nom 🍊), équivaut donc en informatique à... un autre sous-réseau !



Mais que vient faire un masque ici ?

Ce serait très difficile d'expliquer directement cette notion, alors nous utiliser notre formule magique : analogie, magie ! 🧙‍♂️

Dans une adresse postale, il y a un numéro et un nom de rue. Par exemple : 17 rue des Coquelicots (au hasard 🍊). Un masque, c'est ce qui sépare le numéro du nom de la rue. Pour une adresse postale, ça se voit à l'œil nu (on sait reconnaître en un coup d'œil un numéro). Mais en réseau, c'est différent.

Prenons l'adresse IP 10.54.29.84 (au hasard, toujours). On ne peut pas, à première vue, reconnaître le numéro (de l'hôte) de la rue (le réseau, ou sous-réseau) : il n'y a que des chiffres ! C'est pour ça qu'on a recours à un masque : c'est une suite de nombres qui dit que telle partie correspond au nom de la rue (au sous-réseau) et telle partie identifie l'hôte (le numéro de la maison). On verra dans les chapitres suivants comment se représente un masque. 🍊

Prenons un autre exemple : le téléphone (ce n'est pas pour rien qu'on a évoqué le réseau télécom avec le réseau Internet!).

Si vous souhaitez téléphoner, que faites-vous ? C'est simple : vous prenez votre téléphone, vous tapez le numéro de votre correspondant, puis vous validez (en général, parce qu'il y a toujours des téléphones bizarres 🍊). Le numéro de votre correspondant peut, là encore, être assimilé à une adresse IP. 🍊

Cependant, si vous appelez à l'international, comment faire ? Si votre ami habite le Cameroun par exemple, vous devez rentrer l'indicatif national de son pays. Dans notre cas, c'est 237 (vous rentrerez alors +237 sur les portables et 00237 sur les fixes généralement). Vous voyez le rapport avec les sous-réseaux ? Un pays représente dans notre exemple un sous-réseau du réseau télécom mondial et l'indicatif de ce pays est équivalent au masque du sous-réseau.

On voit donc dans ces deux exemples que l'adresse IP (le numéro de la maison ou le numéro de téléphone) appartient à un sous-réseau.

### III. Veuillez vous identifier pour communiquer

En reprenant le parallèle que l'on vient de faire entre un réseau « humain » et un réseau informatique, et maintenant que l'on a tout le vocabulaire, vous devez être capable de transformer les trois lois précédentes en les appliquant à un réseau informatique...

La correction ? La voici :

👁 Contenu masqué n°1

?

Mais alors, comment faire pour que deux machines, appartenant à des sous-réseaux différents, communiquent ?

C'est là qu'intervient...

#### III.1.2.2. ...La passerelle

Celle-ci permet donc la communication entre deux sous-réseaux :

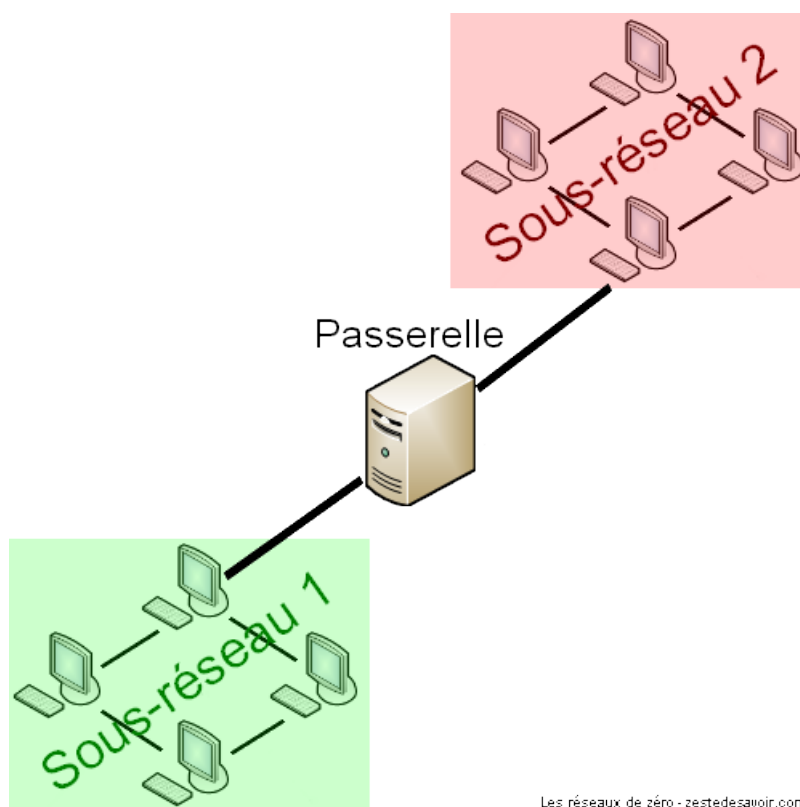


FIGURE III.1.2. – Une passerelle qui relie 2 sous-réseaux entre eux

Une passerelle est un autre ordinateur qui a plusieurs cartes réseau (en général, c'est un routeur). Cet ordinateur peut communiquer avec plusieurs sous-réseaux. On peut le comparer à une personne située à un carrefour, c'est-à-dire un croisement de plusieurs rues. La passerelle sert ainsi de messager entre les habitants des différentes rues. Il faut un peu d'imagination pour comprendre...

i

On parle aussi de passerelle par défaut, de passerelle applicative ou de passerelle logique. Tous ces termes sont synonymes.

Un hôte communique avec la passerelle par défaut selon l'architecture **client-serveur**.

### III.1.3. Le client et le serveur

Client et serveur, voici 2 mots que vous pouvez rencontrer dans la vie courante. Dans un café, par exemple. Un client est une personne qui demande quelque chose au serveur : le client demande un café au serveur, qui lui apporte. En informatique, le principe est le même : un client va demander quelque chose au serveur. Un exemple très simple : quand vous allez sur Zeste de Savoir, vous êtes un client qui demande au serveur du site une page. Dans la théorie, c'est aussi simple que ça.

Le mode de communication entre un client et un serveur est appelé **architecture client-serveur**.

Un autre exemple ? Les serveurs IRC. Pour ceux qui ne connaissent pas, un serveur IRC est un serveur (eh oui 🍊 ) sur lequel des clients peuvent venir discuter (« *chatter* ») sur des salons. Ce système était très populaire jusque dans les années 2000 pour les discussions de groupe (le *chat* de Twitch reposait initialement sur IRC), avant d'être supplanté par d'autres programmes comme Discord. Un des clients ayant rejoint un salon peut envoyer un message au serveur en lui demandant de le transmettre aux autres, ce qu'il s'empresse de faire comme le montre cette animation :

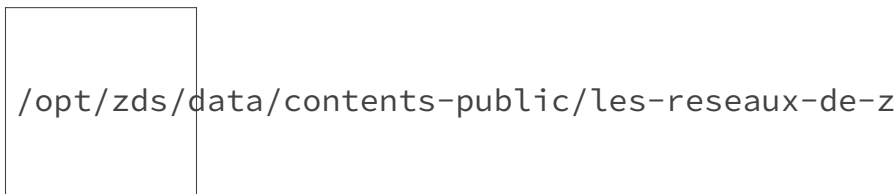


FIGURE III.1.3. – Des clients connectés à un serveur IRC, sur le même salon, s'échangent des messages

i

Bien que cette architecture se retrouve dans beaucoup d'*applications* d'Internet (eh oui, il faut se remémorer le premier chapitre, dur 🍊 ), il existe un autre mode de communication : le pair-à-pair (P2P). Il s'avère très pratique dans certaines applications, comme le partage de fichiers notamment.

## Conclusion

Vous devriez à ce stade comprendre l'identification dans un réseau. Mais n'allez pas vous imaginer que c'est si simple ! Maintenant, on va aller plus en profondeur...

*III. Veuillez vous identifier pour communiquer*

## Contenu masqué

### Contenu masqué n°1

Pour que 2 hôtes (machines connectées) communiquent :

- Ils doivent utiliser le même protocole ;
- Ils doivent appartenir au même sous-réseau ;
- Chaque hôte doit connaître l'adresse IP de l'autre.

[Retourner au texte.](#)

## III.2. Les masques de sous-réseaux : à la découverte du subnetting

### Introduction

Après avoir abordé la notion de masque de sous-réseaux, nous allons voir concrètement de quoi il s'agit. On va commencer à accélérer à partir de ce chapitre, alors soyez attentifs ! 🍌



Ce chapitre sur les masques de sous-réseaux n'est valable que pour les adresses IPv4.

#### III.2.1. En bref

Un masque de sous-réseau, ça ressemble un peu à une adresse IP dans la forme, mais chaque octet ne peut prendre que certaines valeurs. Des exemples : 255.255.0.0, 255.255.255.0,... On les associe à des adresses IP et cela définit une **plage d'adresses** qui vont constituer un réseau. C'est donc le masque qui va définir avec qui on peut communiquer.

Prenons une adresse IP quelconque : 42.51.82.3. Associons à cette adresse un masque, par exemple 255.0.0.0. Ce masque va définir quelle partie de l'adresse IP **identifie le réseau** (cette partie est appelée **network ID**) et quelle partie **identifie l'hôte sur le réseau (host ID)**. C'est bien compris ? Il vaudrait mieux, car nous allons maintenant voir comment cette définition du network ID et de l'host ID se fait. 🍌

#### III.2.2. L'importance des masques

Un masque de sous-réseau définit donc la plage d'adresses IP avec laquelle une carte réseau peut communiquer directement. Pour communiquer avec des adresses IP extérieures à cette plage, elle doit passer par une passerelle par défaut. Il est maintenant temps de voir la relation qui lie cette plage au masque.

##### III.2.2.1. Relation entre network ID et masques

Regardez bien cet exemple d'adresse IP et son masque de sous-réseau associé :

Adresse	42.51.82.3
---------	------------

Masque	255.255.0.0
--------	-------------

Les octets du masque ayant pour valeur 255 sont les mêmes que les octets de l'adresse IP définissant le network ID. De même, les octets du masque valant 0 correspondent aux octets de l'adresse IP définissant l'host ID. L'adresse IP ci-dessus est donc celle d'un **hôte 3.5 dans le réseau 129.51**. Cela est d'une importance capitale, et vous aurez l'occasion de vous en rendre compte quand nous verrons la personnalisation des masques. Avant d'introduire cette notion, voyons d'abord...

### III.2.2.2. Des règles fondamentales à connaître absolument

Un masque de sous-réseau ne peut pas s'écrire n'importe comment. Voici quelques règles à connaître par cœur avant même d'aller plus loin :

**On ne peut pas mélanger les zéros et les autres valeurs.** En somme, tous les 255 doivent être à gauche et les zéros à droite. Pourquoi ? Parce que dans une adresse IP, c'est la partie gauche qui correspond à l'identité du réseau, et la partie droite qui correspond à l'identité de l'hôte. Ces exemples de masques sont donc **invalides** : 255.0.0.255, 255.255.0.255, 0.0.0.255,...

**Un masque de sous-réseau ne peut pas avoir un octet qui vaut plus de 255**, pour la bonne et simple raison qu'un octet ne peut prendre que 256 valeurs différentes, ici de 0 à 255. Par conséquent, un masque de sous-réseau ne peut pas prendre de valeur négative.

Ces règles sont simples, mais il faut absolument les savoir pour aller plus loin dans l'étude des masques de sous-réseau. Nous aurons l'occasion d'en voir d'autres par la suite, notamment lors de l'étude du *subnetting*.

### III.2.3. Introduction au subnetting

Le *subnetting* est une technique qui consiste à diviser un réseau plus large en plusieurs sous-réseaux. Décomposons ce mot :

sub - net - ting sous - réseau - (suffixe d'action)

Il n'existe apparemment pas d'équivalent français. Si vous avez envie de dire « sous-réseautage », libre à vous, mais on risque de vous regarder bizarrement... 🍊 Vous l'aurez peut-être deviné, le *subnetting* est l'action de créer des sous-réseaux. Et pas n'importe comment : en **personnalisant les masques**.

Par exemple, admettons un réseau de 1000 ordinateurs. La gestion d'un tel réseau ne doit pas être évidente. Grâce au *subnetting*, on peut par exemple **diviser** ce grand réseau en 10 réseaux de 100 ordinateurs chacun (en gros). Et cela procure des avantages, voyez par vous-même !

### III.2.3.1. Délégation de l'administration

Le *subnetting* permettant de diviser un grand réseau en plusieurs réseaux plus petits, il permet de décentraliser l'administration, et éventuellement de déléguer la gestion de chaque sous-réseau à une personne différente. Dans une entreprise possédant un réseau de 1000 machines, sa gestion sera simplifiée.

### III.2.3.2. La réduction du trafic

Si 2 ordinateurs se trouvant dans un même sous-réseau communiquent, ils n'exploiteront que la bande passante allouée à leur sous-réseau, et non celle du réseau entier. Considérons une entreprise possédant un réseau de 500 machines. Il est divisé en 25 sous-réseaux de 20 machines. Ainsi, les machines appartenant à un même sous-réseau communiquant entre elles n'utilisent que la bande passante qui est allouée à leur sous-réseau, ce qui permet de ne pas réduire le débit des autres. Cela se remarque notamment lors du *broadcast* de données, c'est-à-dire l'envoi à tous les hôtes : elles ne sont transmises qu'aux ordinateurs du sous-réseau, et pas aux autres qui n'en ont probablement rien à faire.

### III.2.3.3. La facilité du diagnostic

Si par exemple un ordinateur consomme une quantité de bande passante inhabituelle, il est beaucoup plus aisé d'analyser son comportement pour régler le problème lorsqu'il se trouve dans un petit sous-réseau que lorsqu'il se trouve dans le même réseau que 1000 autres machines. C'est encore un avantage.

### III.2.3.4. L'économie d'adresses

Prenons par exemple une adresse IP : 200.10.0.5. Le masque de sous-réseau par défaut est 255.255.255.0. Dans ce cas, on peut avoir jusqu'à 254 terminaux (clients) dans ce même réseau, donc 254 adresses IP. Ce qui veut dire que si vous avez un réseau de 10 ordinateurs, vous avez quand même 254 adresses IP disponibles. Mais comme vous ne les utilisez pas, vous les **gaspillez**. Toutefois, le *subnetting* ne nous permet pas d'économiser comme on le souhaite.



Vous avez 254 adresses IP disponibles uniquement lorsque vous utilisez un masque de sous-réseau par défaut.



Et ça sert à quoi d'économiser des adresses IP ? Ça ne va pas coûter plus cher de laisser 200 adresses IP vacantes, que d'en laisser 2...

Dans un réseau privé, certes. Mais cela peut être utile pour des raisons de sécurité, entre autres. Nous ne pouvons pas encore voir réellement l'intérêt, vous vous en rendrez compte en temps voulu. Sachez toutefois que sur Internet, les adresses IP publiques s'achètent. Et ce n'est pas le même prix d'acheter 2 adresses que 200 (ça doit faire environ 100 fois plus 🍊 ).

?

Donc le *subnetting* permet de diviser un réseau en plusieurs sous-réseaux, ça a plein d'avantages, mais ça se met en place comment, concrètement ?

C'est le sujet du prochain chapitre ! Hé oui, « introduction au *subnetting* », ça veut dire « définition et du blabla » ! Vous croyiez quoi ? Que vous alliez « subnetter » sans savoir à quoi ça sert, juste pour dire « je suis trop fort, j'ai subnetté mon *home network* » ? 🍊

Avant de subnetter, voici des informations qui vous seront probablement utiles, notamment si vous débutez en tant qu'administrateur réseau.

### III.2.4. Analyse des contraintes et plan d'adressage

Vous vous en doutez peut-être, les administrateurs réseaux passent beaucoup plus de temps à analyser qu'à implémenter. C'est d'ailleurs le rôle principal d'un administrateur réseau : apporter son expertise dans l'analyse et le design d'une infrastructure réseau. Quant à l'implémentation, c'est relativement simple une fois l'analyse terminée. En fait, c'est comme en programmation. Il y a le chef de projet qui analyse les contraintes et les demandes des clients, écrit éventuellement un cahier des charges, et le remet aux développeurs qui se serviront des contraintes de ce dernier pour créer une application. En réseau, c'est le même principe : vous, l'administrateur, allez réfléchir sur les contraintes du réseau, et vous allez proposer une solution en tenant compte de plusieurs critères (le prix, la facilité de mise en place, l'évolution de l'infrastructure, etc.).

#### III.2.4.1. Analyse des contraintes

Avant de « subnetter » un réseau, il faut donc faire une minutieuse analyse. Nous allons vous donner quelques pistes.

#### III.2.4.2. Le prix

Subnetter un réseau, c'est le subdiviser en plusieurs sous-réseaux. Ceci dit, il en résulte explicitement que l'achat de matériel additionnel est obligatoire, en effet il faudra un routeur pour que les sous-réseaux obtenus puissent communiquer. Qui dit nouveau matériel dit... câblage. 🍊 Bref, il faut prendre en compte cette contrainte financière. Un client (ou votre patron) peut vous spécifier un budget pour l'infrastructure à mettre en place et il faudra trouver un compromis pour allier « meilleur prix » et « meilleure solution », ce n'est pas toujours évident, les boss sont trop exigeants. Parfois. 🍊

#### III.2.4.3. L'évolution du réseau

Un bon administrateur n'est pas celui qui offre une solution idéale à court terme. Les réseaux sont un domaine de l'informatique qui évolue très vite. Il ne faut jamais penser à une solution qui ne serait fonctionnelle que pendant 1 an. Il est préférable se poser la question : « dans 2-3

### III. Veuillez vous identifier pour communiquer

ans, à quoi ressemblera mon réseau ? Sera-t-il facile d'évoluer vers une nouvelle infrastructure si j'utilise telle infrastructure ? ».

#### III.2.4.4. Le nombre d'adresses IP

Il faut déterminer le nombre d'adresses IP dont on aura besoin. Les administrateurs débutants ont tendance à choisir pile-poil un sous-réseau qui leur offre exactement le nombre d'adresses IP dont ils ont besoin (c'est rare mais c'est néanmoins possible). Or cette pratique est une erreur, ou du moins, elle est fortement déconseillée. Si nous nous restreignons à un sous-réseau qui nous permet d'avoir 17 adresses IP par exemple, et que dans un futur proche nous ajouterons 400 autres ordinateurs... Vous rendez-vous compte de l'ornière dans laquelle nous nous trouverons ? Il faudra re-subnetter correctement et redéfinir les plages, et c'est... ennuyeux. 🍊

i

Il est recommandé de choisir un masque en se basant sur le maximum d'adresses IP qu'un réseau donné pourrait avoir, et non le minimum ou l'actuel. Par exemple, si vous avez besoin d'un sous-réseau de 10 adresses IP et qu'il peut y avoir agrandissement de réseau, que vous êtes sûrs que ça ne dépassera pas un maximum de 40 ordinateurs, il serait alors judicieux de commencer par choisir un masque qui vous donne d'ores et déjà 40 adresses IP. Cela peut être considéré comme du gâchis d'adresses mais c'est néanmoins pratique pour l'évolution. 🍊

#### III.2.4.5. L'organisation

L'une des choses les plus importantes, hormis les contraintes évoquées ci-dessus, est l'organisation du plan d'adressage.

i

Un plan d'adressage est un plan résultant d'une analyse de contrainte, qui servira de modèle pour gérer l'adressage / l'assignation des adresses dans un réseau donné.

« Comment allez-vous organiser vos sous-réseaux ? » Telle est la question qu'il faut se poser, niveau organisation. Plusieurs méthodes d'organisation sont courantes.

##### III.2.4.5.1. L'organisation par bâtiment

Certaines entreprises ont une organisation par architecture (physique). Par exemple, elles peuvent avoir le bâtiment A, qui regroupe le *staff* se chargeant du service après-vente. Elles peuvent également avoir le bâtiment C, qui regroupe le *staff* se chargeant du service des finances, etc. Vous pouvez par conséquent être amené à subnetter et organiser les sous-réseaux par bâtiment : créer un sous-réseau pour le bâtiment A, un autre pour le bâtiment B, etc. Donc cette organisation consiste à créer autant de sous-réseaux qu'il y a de bâtiments. 🍊 Elle a ses avantages, car elle offre une facilité de diagnostic grâce au repérage physique. Par exemple, on pourrait facilement dire que l'ordinateur D02 qui a des difficultés à communiquer est localisé dans le bâtiment D. C'est donc une méthode d'isolation utile en cas de diagnostic. 🍊

i

Dans ce genre d'organisation, les hôtes sont souvent nommés par un motif : nom du bâtiment + numéro d'hôte. Par exemple l'hôte 2 dans le bâtiment H serait nommé H02 (non, ce n'est pas une molécule 🍊).

#### III.2.4.5.2. L'organisation par fonctions

Cette organisation est différente de la précédente. On peut avoir un bâtiment B qui regroupe des employés du service de support informatique. Dans ce bâtiment, il y aura par exemple un chef de projet, des réceptionnistes et des techniciens. Mais il se peut que l'entreprise ait aussi des techniciens en électronique dans le bâtiment C, ou un chef de projet dans le bâtiment D qui s'occupe de la recherche. Dans ce genre de cas, vous pouvez alors subnetter par fonctions. C'est-à-dire, créer un sous-réseau qui n'hébergera **que** les ordinateurs des chefs de projets (tous services confondus), un sous-réseau qui n'hébergera que les secrétaires (tous services confondus), etc.

?

Ouh là, c'est pas de la ségrégation ça ? 🍊

Que nenni. 🍊 Cette organisation peut être très pratique. Imaginez que vous ayez plusieurs techniciens en informatique industrielle, qui communiquent constamment avec un serveur d'applications dans leur domaine. Les logiciels hébergés par le serveur sont lourds, et lorsque tous les techniciens travaillent à un rythme fou et multiplient les requêtes vers le serveur, cela ralentit le réseau. Avec une organisation par fonctions, vous aurez un sous-réseau alloué aux techniciens en informatique industrielle qui implémentera un débit assez élevé, uniquement pour assurer cette fonction. C'est pratique, on peut alors allouer une bande passante précise par sous-réseau en fonctions des contraintes. Car, avouons-le, ça sert à rien d'allouer 512 Mo/s de débit aux secrétaires. 🍊 (Ah, on nous dit dans l'oreillette qu'on va se faire taper par des secrétaires fâchées. On finit le chapitre et on met les voiles ! 🍊)

#### III.2.4.5.3. L'organisation par architecture

Le titre est assez évocateur, donc nous allons faire court (aussi parce que nous manquons d'inspiration 🍊). Cette organisation consiste à subnetter avec une organisation par architecture. Dans la partie I du cours, souvenez-vous, nous avons parlé de la topologie logique « Token Ring ». Grâce à une organisation par architecture, vous pouvez créer un sous-réseau spécial Token Ring, un autre sous-réseau spécial Ethernet, et un autre spécial Wi-Fi, etc. 🍊

Voilà, nous avons fait le tour des techniques d'organisation. Cette phase d'analyse ne vous servira à rien en tant qu'étudiant, cependant quand vous entrerez dans le monde actif en réseau, elle vous sera d'une grande utilité. Et même en stage, ça peut servir... à impressionner le maître de stage ! 🍊 (Assurez-vous quand même auparavant que le « m'as-tu vu » ne l'agace pas!)

*III. Veuillez vous identifier pour communiquer*

## Conclusion

Le prochain chapitre sera donc dédié à la personnalisation des masques de sous-réseau, ce qui permet de faire du *subnetting*. Et par conséquent, de restreindre la commande à distance de la machine à café aux autres. 🍊

## III.3. Le subnetting en pratique

### Introduction

Maintenant que vous savez ce qu'est le *subnetting*, nous allons voir comment cela se fait. Faites chauffer vos méninges, cela demande du calcul ! ... Hé ne partez pas ! C'est facile, vous allez voir ! Restez, voyons ! 🍊



Pour pouvoir suivre ce chapitre, le binaire doit vous être familier. Si tel n'est pas le cas, [consultez cette annexe](#) avant de poursuivre votre lecture.

### III.3.1. Comment ?

Considérant que vous maîtrisez les conversions d'adresses IP du binaire au système décimal et vice versa, que vous connaissez les puissances de deux et que les yeux fermés vous pouvez dire que  $2^3 = 8$ , que vous savez ce qu'est le *subnetting*, et que vous comprenez parfaitement cette notion d'économie d'adresses, nous allons voir **comment** « **subnetter** », comment avoir un masque de sous-réseau personnalisé.



Cette section n'est que théorique, elle consiste simplement à vous dire comment on fait, et dans la suite de ce chapitre nous allons subnetter ensemble, pas à pas.

#### III.3.1.1. Le comment du pourquoi

Pour personnaliser les masques de sous-réseau, il faut **emprunter** des *bits* de la partie host (client) de notre adresse IP. Revoyons d'abord en douceur ce que sont le host ID et le network ID. D'abord, host ID signifie *identité de l'hôte* et network ID signifie *identité du réseau*. La règle de base à retenir est : plus on « monte » dans les masques (c'est-à-dire qu'on passe de 255.0.0.0 à 255.255.0.0, de 255.255.0.0 à 255.255.255.0,...), plus le network ID devient grand : il gagne un octet de plus à chaque fois, et le host ID en perd un.

Faire du *subnetting*, c'est subdiviser un réseau en plusieurs sous-réseaux, diminuer le nombre d'adresses IP par sous-réseau.

Avec le masque 255.0.0.0, le premier octet correspond à l'identité du réseau. Avec 255.255.0.0, ce sont les deux premiers octets, et avec 255.255.255.0, les 3 premiers. C'est l'inverse pour l'identité de l'hôte :

### III. Veuillez vous identifier pour communiquer

Masque de sous-réseau	Adresse IP
255.0.0.0	<b>75.10.2.4</b>
255.255.0.0	<b>135.5.0.7</b>
255.255.255.0	<b>220.42.3.6</b>

En **gras** : network ID, en *italique* : host ID

Donc, pour subnetter un réseau en plusieurs sous-réseaux, on se sert des *bits* disponibles du masque de sous-réseau, c'est-à-dire ceux qui valent 0.

Il est possible de procéder de plusieurs manières pour subnetter :

- En partant du nombre de sous-réseaux désirés ;
- En partant du nombre d'adresses IP désirées par sous-réseau ;
- En combinant les deux, c'est-à-dire en prenant en compte le nombre de sous-réseaux désirés et le nombre d'adresses IP désirées par sous-réseau.

Nous ne verrons pas tout de suite cette dernière possibilité, nous allons voir seulement les 2 premières. Ces méthodes sont valables pour toutes les adresses.

#### III.3.2. À partir du nombre de sous-réseaux désirés

Avant de prendre un exemple concret, vous devez connaître quelques généralités. Tout d'abord, voici une formule à savoir :

$$S = 2^n$$

Dans cette formule, S est le nombre de sous-réseaux désirés. À partir de S, vous devez déterminer n, qui est un nombre entier positif, et qui correspond au nombre de *bits* devant être **masqués**.



Masqués ? Comme au bal ?

Pas vraiment... 🍌 Masquer signifie, en gros, le mettre à 1 pour les besoins du *subnetting*. Comme vous le savez normalement, le *subnetting* consiste en l'emprunt des *bits* de la partie hôte pour créer des sous-réseaux. Ce processus d'emprunt de *bits* est appelé **masquage**. Masquer un *bit*, c'est donc l'emprunter, l'allumer.

En fait, S ne pourra pas toujours être égal au nombre de sous-réseaux désirés, nous verrons dans les exemples comment nous allons faire dans ce cas très courant.



Autrefois, dans ce cours, nous montrions une formule légèrement différente :  $S = 2^n - 1$ . Ce « -1 » venait d'une convention selon laquelle l'octet de l'adresse IP définissant le sous-réseau ne pouvait être supérieur ou égal à l'octet modifié du masque de sous-réseau



personnalisé. Par exemple, si on avait un réseau 198.15.2.0 et qu'on applique aux hôtes un masque 255.255.255.192, on ne pouvait pas avoir de sous-réseau ayant pour identité 198.15.2.192. Cette pratique étant dépassée, nous ne la prenons plus en compte dans ce cours. Sachez que vous pourriez encore tomber sur de vieux routeurs qui vont vous casser les pieds avec ça.

Au vu de vos têtes sceptiques (enfin, on ne voit pas mais on devine 🍊), nous allons tout de suite faire un exemple, parce que la théorie, ça ne semble pas digeste.

### III.3.2.1. Exemple de subnetting

Passons tout de suite au plus intéressant. Considérons le réseau **40.0.0.0** avec pour masque global **255.0.0.0**. Nous voulons le diviser en 20 sous-réseaux. Déterminons le nombre de *bits* à masquer pour obtenir un masque de sous-réseau personnalisé, qui devra être appliqué à tous les hôtes.



Ah oui, on ne peut pas obtenir exactement 20 avec la formule  $2^n$  ! Dans ce cas, nous allons prendre une valeur un peu supérieure pour avoir au moins 20 sous-réseaux. Allons-y doucement :

- $2^4 = 16$
- $2^5 = 32$
- On a suffisamment de réseaux en masquant 5 *bits*, on arrête là.



Quand vous avez trop d'adresses IP par sous-réseau, vous devriez empêcher l'assignation des adresses inutilisées. C'est une question de sécurité, pour empêcher qu'un intrus puisse s'octroyer une adresse IP libre. Nous n'entrerons pas dans les détails ici.

On ne peut pas mélanger les 1 et les 0, **tous** les 1 doivent être à gauche, et les 0 à droite. Cela veut dire le masquage se fait de la gauche vers la droite.

Nous allons donc masquer 5 *bits* de cette manière. Nous avons les puissances suivantes (les 8 premières puissances de 2) :

- $2^7 = 128$
- $2^6 = 64$
- $2^5 = 32$
- $2^4 = 16$
- $2^3 = 8$
- $2^2 = 4$
- $2^1 = 2$
- $2^0 = 1$

### III. Veuillez vous identifier pour communiquer

Voilà donc les 8 premières puissances de deux, par ordre décroissant. Nous allons masquer les 5 *bits* qu'il nous faut pour obtenir 20. Donc, nous allons **additionner** les valeurs des 5 premières puissances ci-dessus. Cela nous donne donc :  $2^7 + 2^6 + 2^5 + 2^4 + 2^3 = 248$ .

Nous avons masqué 5 *bits* du 2ème octet de notre masque de sous-réseau. Schématiquement, ça nous donne ceci :

255	248	0	0
ssssssss	sssshhhh	hhhhhhhh	hhhhhhhh

$s = subnet ; h = host$

La valeur de notre **nouveau masque de sous-réseau** est à présent **255.248.0.0**.

Si vraiment vous n'avez pas compris comment le 248 a été obtenu :

👁 Contenu masqué n°2

Et voilà, soyez très heureux, nous avons réussi à personnaliser un masque de sous-réseau ! 🍊

Maintenant il faudrait définir les limites de chaque sous-réseau, sinon ça ne va pas être très utile. Dans le cas présent, nous en avons 31, on va vous donner les 5 premières plages et la dernière, vous allez faire le reste vous-mêmes, il faut bien que vous fassiez quelque chose ! 🍊

?

Comment calculer les plages ?

C'est relativement simple. Pour calculer les plages, il faut retrancher le nombre calculé du nouveau masque de sous-réseau à 256. Ce nombre est 248,  $256 - 248 = 8$ . Donc nos sous-réseaux seront séparés par un intervalle de 8.

Concrètement, reprenons le réseau que nous avons choisi à la base : 40.0.0.0. Le premier sous-réseau est 40.0.0.0, le deuxième est 40.8.0.0, le troisième 40.16.0.0, le quatrième 40.24.0.0, etc. 🍊

Pour calculer les plages, il faut savoir que la dernière adresse d'un sous-réseau donné est toujours égale à l'adresse de l'identité du prochain sous-réseau moins 1. 🍊 Un exemple concret ? Dans notre cas, notre premier sous-réseau est 40.0.0.0. La première adresse IP **adressable** (pouvant être donnée à un hôte) est donc 40.0.0.1 et la dernière... 40.7.255.254. 🍊

!

Techniquement, la dernière adresse dans un réseau est réservée pour la diffusion dite de *broadcast*. Cela sert à envoyer la même chose à tous les hôtes du réseau. Ici, l'adresse 40.7.255.255 est réservée pour le *broadcast* et n'est pas donc pas assignable à un hôte. Nous y reviendrons un peu plus tard.

Nous avons donc :

### III. Veuillez vous identifier pour communiquer

Ordinal	Adresse du sous-réseau	Première adresse IP d'hôte	Dernière adresse IP d'hôte
1er	40.0.0.0	40.0.0.1	40.7.255.254
2ème	40.8.0.0	40.8.0.1	40.15.255.254
3ème	40.16.0.0	40.16.0.1	40.23.255.254
4ème	40.24.0.0	40.24.0.1	40.31.255.254
5ème	40.32.0.0	40.32.0.1	40.39.255.254
...	...	...	...
Dernier	40.248.0.0	40.248.0.1	40.255.255.254

Vous êtes capables de faire le reste maintenant. 🍊

Si vous pouvez encore suivre, nous allons voir comment subnetter à partir du nombre d'adresses IP d'hôtes désiré. N'hésitez pas à faire une pause si vous pensez en avoir déjà beaucoup fait pour le moment.

### III.3.3. À partir du nombre d'adresses d'hôtes désirées

Parés à affronter cette méthode ? Dans ce cas, voici une bonne et une mauvaise nouvelle. On commence par la mauvaise ? La formule et la méthode changent ! La bonne nouvelle ? Elles ne changent que très peu, et vous allez comprendre pourquoi !

#### III.3.3.1. Explications sur l'adresse de broadcast et l'identité du réseau

La nouvelle formule est :

$$S = 2^n - 2$$

Cette fois, S correspond au nombre d'hôtes désiré par sous-réseau. La raison du changement dans la formule est simple : on retranche une première unité pour l'identité du réseau car elle n'est pas assignable. Si l'adresse 40.16.0.0 identifie un réseau, elle ne peut pas identifier un hôte ! Une autre unité est retranchée car on ne peut pas non plus assigner l'adresse de *broadcast*.

i

Il est théoriquement possible de supprimer le *broadcast* dans un réseau, mais dans des cas très particuliers uniquement, que nous ne verrons pas ici. Il est indispensable dans la plupart des cas : par exemple, les requêtes ARP permettant d'établir des correspondances entre adresses IP et MAC sont envoyées en *broadcast* ! Si vous décidiez de la supprimer et d'utiliser cette adresse pour un hôte, la formule deviendrait  $2^n - 1$ .

### III. Veuillez vous identifier pour communiquer

Ces explications sont facilement vérifiables lorsqu'on détermine les plages d'un réseau subnetté avec le masque 255.255.255.0, vous aurez l'occasion de le voir. Pour le moment, voyons une application de cette méthode avec un exemple.

#### III.3.3.2. Un autre exemple de subnetting

Prenons le réseau 158.37.0.0. Commençons par décider du nombre d'adresses IP que l'on souhaite avoir par sous-réseau.



Si vous choisissez un nombre inférieur à 255, vous vous retrouverez avec un masque sous la forme 255.255.255.xxx. Ce n'est pas interdit, ça reste du *subnetting*, mais ne soyez pas surpris si le dernier octet du masque est également modifié.

Considérons que nous voulons 1800 hôtes par sous-réseau. Déterminons  $n$  :

- $2^{10} - 2 = 1022$
- $2^{11} - 2 = 2046$

$n$  vaut donc 11. C'est là que ça change : comme nous voulons un nombre précis d'adresses par sous-réseaux, 11 *bits* doivent être **libres pour les hôtes** ! Ce qui signifie que 11 *bits* doivent valoir 0. Il y a 32 *bits* par masque, pour connaître le nombre de *bits* devant valoir 1, on fait  $32 - 11 = 21$ . Notre nouveau masque doit donc comporter 21 *bits* allumés, écrivons-le en binaire :

11111111 . 11111111 . 11111000 . 00000000

Ce qui nous donne en décimal **255.255.248.0**. L'intervalle entre chaque sous-réseau est de  $256 - 248 = 8$ .

Dressons maintenant le tableau des plages :

Ordinal	Adresse du sous-réseau	Première adresse IP d'hôte	Dernière adresse IP d'hôte
1er	158.37.0.0	158.37.0.1	158.37.7.254
2ème	158.37.8.0	158.37.8.1	158.37.15.254
3ème	158.37.16.0	158.37.16.1	158.37.23.254
...	...	...	...
Dernier	158.37.248.0	158.37.248.1	158.37.255.254

Voilà donc un certain nombre de sous-réseaux avec 2046 adresses d'hôtes dans chaque. On n'en voulait que 1800, mais ce n'était pas possible de les avoir précisément, donc on a pris la valeur possible immédiatement supérieure.

Faisons maintenant un autre exemple, mais cette fois, il doit y avoir moins de 254 hôtes par sous-réseau. La méthode reste la même, mais nous allons voir quelques particularités, dont une qui permet de vérifier facilement la formule de départ.

### III.3.3.3. Exemple de subnetting avec moins de 254 hôtes par sous-réseau

Procédons de la même manière. Dans le réseau 203.68.5.0, nous voulons 14 hôtes par sous-réseau.

$2^4 - 2 = 14$  ! Super, on tombe juste sur 14 ! (Bon d'accord, c'était fait exprès pour éviter de chercher... 🍊 )

Attention tout de même quand ça tombe juste comme ça... Si par malheur vous deviez rajouter un hôte dans un sous-réseau, vous seriez ~~dans la m~~ bien embêté car vous devrez reconfigurer **toutes les machines du réseau** !

On a donc  $n = 4$ , il nous faut 4 *bits* valant zéro. Comme ici on ne va modifier que le dernier octet, on peut faire directement  $8 - 4 = 4$  pour connaître sa nouvelle valeur.  $11110000_{(2)} = 240_{(10)}$ , notre nouveau masque est donc **255.255.255.240**. L'intervalle est de  $256 - 240 = 16$ , on détermine les plages :

Ordinal	Adresse du sous-réseau	Première adresse IP d'hôte	Dernière adresse IP d'hôte
1er	203.68.5.0	203.68.5.1	203.68.5.14
2ème	203.68.5.16	203.68.5.17	203.68.5.30
3ème	203.68.5.32	203.68.5.33	203.68.5.46
...	...	...	...
Dernier	203.68.5.240	203.68.5.241	203.68.5.254

Vous remarquez probablement une différence : la dernière adresse IP d'hôte de chaque sous-réseau ne se termine pas par 254 ! De plus, vous voyez bien maintenant l'intérêt du -2 de la formule : l'adresse du réseau et celle de *broadcast* sont bien visibles ici.

Remarquez que le masque de sous-réseau ne peut être 255.255.255.255. En effet, dans ce cas, il n'y a que l'adresse même du sous-réseau dans chaque sous-réseau, donc aucune adresse disponible pour les hôtes ! D'ailleurs, si on prend comme masque 255.255.255.254, il n'y a qu'une adresse disponible par sous-réseau, on est donc obligé de supprimer le *broadcast* (ce qui n'est pas grave vu qu'il n'y a qu'un hôte).

Allez, vous avez bien travaillé, vous avez droit... à la suite ! 🍊

### III.3.4. La notation du masque

Cette sous-partie ne comportera rien de fameux, nous allons juste vous fournir quelques explications sur les éventuelles notations que vous rencontrerez probablement dans le monde du réseau.

### III.3.4.1. La notation « classique »

Cette notation dite « classique » est la notation « normale » d'une adresse IP. C'est en fait une notation qui couple l'adresse IP et son masque de sous-réseau associé. Par exemple, vous pourrez rencontrer une expression telle que **192.168.1.45/255.255.255.0**. C'est assez évident à comprendre, n'est-ce pas ? Cela veut simplement dire qu'à l'adresse IP 192.168.1.45 est attribué un masque 255.255.255.0. C'est une notation que nous pourrions qualifier de « obsolète » car elle a laissé sa place à...

### III.3.4.2. La notation avec un slash (/)

Cette notation suit le même modèle que la notation classique. C'est-à-dire que c'est un couplage de l'adresse IP d'un hôte à son masque de sous-réseau. Mais le point particulier ici, c'est qu'au lieu de donner l'expression « brute » du masque de sous-réseau dans la notation, on se contente de spécifier le nombre de *bits* masqués pour obtenir ce masque. La notation précédente en notation avec un slash devient **192.168.1.45/24**. Cela veut dire que l'adresse IP 192.168.1.45 est associée à un masque ayant 24 *bits* de masqués.

La notation avec un slash semble devenir la plus courante et la plus utilisée aujourd'hui notamment avec le succès du CIDR (*Classless Inter Domain Routing*) que nous allons aborder très bientôt. En fait, la notation avec un slash n'est rien d'autre que ce qu'on appelle officiellement la **notation CIDR**. 🍊



Concernant les expressions « notation classique » et « expression brute », ce sont des expressions propres au tutoriel, n'allez pas croire qu'elles sont conventionnées. 🍊

Voilà, vous savez tout sur les notations que vous pouvez rencontrer. Il va sans dire que vous préférerez sûrement utiliser la notation avec un slash. C'est plus pratique : on peut se contenter d'écrire 130.14.56.3/16 par exemple au lieu de 130.14.56.2/255.255.0.0. 🍊

## Conclusion

Reposez-vous avant de passer au chapitre suivant. Car vous allez encore faire des calculs, d'un autre genre cette fois ! 🍊

## Contenu masqué

### Contenu masqué n°2

- $2^7 = 128$
- $2^6 = 64$
- $2^5 = 32$
- $2^4 = 16$

*III. Veuillez vous identifier pour communiquer*

- $2^3 = 8$
- $128 + 64 = 192$
- $192 + 32 = 224$
- $224 + 16 = 240$
- $240 + 8 = 248$

[Retourner au texte.](#)

## III.4. La passerelle : les bases du routage

### Introduction

Vous vous êtes remis de vos émotions, ou plutôt, des calculs du chapitre précédent ? Tant mieux ! Ou tant pis pour vous, en cas de réponse négative. Voici un chapitre expliquant le fonctionnement d'une passerelle, avec comme promis, des calculs ! Ne cherchez pas à vous enfuir, cette pièce vient d'être déconnectée du réseau, il n'y a plus de route derrière... Mwahahahaha ! 🙈 (Qui a dit qu'on était des détraqués ? 🍊 )

#### III.4.1. Une petite révision

Deux hôtes ne se situant pas dans le même sous-réseau ne peuvent pas communiquer directement. Il faut que quelque chose intervienne entre les deux pour transmettre à l'un, les données au nom de l'autre. Ce « quelque chose » est la **passerelle** : un service qui connecte plusieurs sous-réseaux. Cette position fait donc que la passerelle se situe à califourchon entre plusieurs sous-réseaux, faisant ainsi office d'intermédiaire.



Ceci est une notion logique, c'est un service fourni par un matériel. En pratique, ce matériel est presque toujours un routeur.

À ce stade du cours, vous ne devriez plus avoir besoin d'analogie !

Dans un réseau comprenant plusieurs routeurs, on rencontre parfois une passerelle par défaut (*default gateway*, en anglais). Passerelle, car elle interconnecte des sous-réseaux, « par défaut » car c'est vers elle qu'on se tourne quand on ne sait pas à qui s'adresser. Chaque routeur a une **table de routage**. Nous avons défini précédemment que les **paquets** représentent des parties de vos données. D'un point de vue réseau, lorsque vous envoyez des données, celles-ci sont découpées en plusieurs portions et chaque portion est appelée un paquet. Quant à la table de routage, il s'agit d'une liste des différentes « routes » (chemins) vers d'autres sous-réseaux. Ces définitions sont très simplifiées pour vous permettre de comprendre le principe, nous reviendrons dessus plus tard. 🍊

C'est bien beau tout ça, mais comment fonctionne cette fameuse passerelle ? La réponse vous attend sagement dans la sous-partie suivante. 🍊

### III.4.2. Mode de fonctionnement

Prenons un exemple concret pour illustrer le mode de fonctionnement d'une passerelle. Voici 2 ordinateurs : Azur-PC et Safran-PC (on se demande bien d'où ces noms sont inspirés 🍊). Leurs cartes réseau sont configurées ainsi :

Nom	Adresse IP	Masque de sous-réseau
Azur-PC	192.0.1.5	255.255.255.0
Safran-PC	72.40.2.1	255.0.0.0

Ils n'appartiennent pas au même sous-réseau et ne peuvent donc pas communiquer : il leur faut une passerelle. Voyons comment elle fonctionne.

#### III.4.2.1. Le fonctionnement

Azur-PC a recours à un processus nommé *ANDing*, que nous allons voir juste après, pour déterminer si Safran-PC, avec qui il veut communiquer, est dans le même sous-réseau que lui. Il réalise que ce n'est pas le cas, il va donc transférer son message à la passerelle en lui indiquant l'adresse du destinataire. Supposons que ce soit un routeur qui offre ce service. Il a 2 interfaces. Pour que la communication puisse avoir lieu, une de ses interfaces doit être dans le même sous-réseau que Azur-PC et l'autre dans le même que Safran-PC. Voici une configuration possible pour ce routeur :

Interface	Adresse IP	Masque de sous-réseau
A	192.0.1.6	255.255.255.0
B	72.40.1.1	255.0.0.0

Avec une telle configuration, Azur-PC et Safran-PC peuvent à présent communiquer. Quand Azur-PC voudra parler à Safran-PC, il vérifiera grâce au *ANDing* si le destinataire est dans le même sous-réseau. Si oui, il enverra son message directement à son adresse IP, sinon, il l'envoie à la passerelle en lui demandant de transmettre à bon port. La passerelle étant entre les 2, cela ne pose pas de problème.

Voici un schéma récapitulatif :

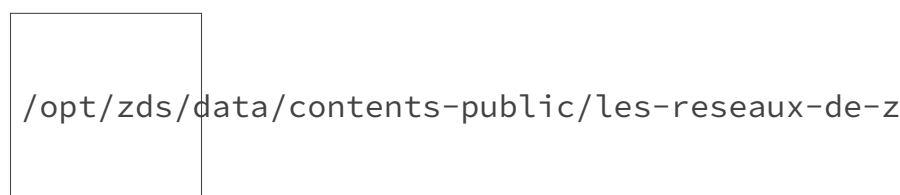


FIGURE III.4.1. – C'est peut-être moche mais faites comme si ce n'était pas le cas. :-°


Il est maintenant temps de voir ce qu'est le *ANDing* dont on a parlé sans expliquer ce que c'était.

### III.4.3. ANDing (conjonction logique)

Dans la sous-partie précédente, nous avons évoqué un processus utilisé par Azur-PC pour déterminer dans quel sous-réseau se trouve le destinataire : le *ANDing*.

#### III.4.3.1. Le ANDing, approche théorique

##### III.4.3.1.1. Définition

Le *ANDing* ou conjonction logique (ET logique) en français, est utilisé pour le [calcul des propositions](#) . Nous n'allons pas entrer dans les détails de ce que c'est concrètement. Nous allons simplement voir en quoi cela peut nous servir dans les réseaux en général, mais surtout quelle est son application dans le cas du routage. Quand votre hôte veut communiquer, il fait un calcul logique pour déterminer si votre destinataire se trouve dans un même sous-réseau ou pas. Si le destinataire est dans un sous-réseau différent, les données seront envoyées à la passerelle qui se chargera de les *router* au destinataire.

##### III.4.3.1.2. Les règles du ANDing


C'est là que l'on voit l'intérêt d'être à l'aise avec la conversion des adresses IP du décimal au binaire. En effet, le *ANDing* se base sur la notation binaire des adresses IP. Si vous n'êtes pas à l'aise avec les conversions décimal/binaire, nous vous conseillons de lire l'annexe qui traite ce sujet. Le *ANDing* est relativement simple à comprendre, il faut juste assimiler les règles suivantes :

- $0 \text{ AND } 0 = 0$  ;
- $0 \text{ AND } 1 = 0$  ;
- $1 \text{ AND } 0 = 0$  ;
- $1 \text{ AND } 1 = 1$ .

Facile non ? 

?

Oui, mais concrètement, à quoi servent ces règles ?

Déterminer si l'adresse IP du destinataire est dans le même sous-réseau que celle de l'émetteur est assez simple. La carte réseau de l'émetteur connaît son adresse IP, son masque de sous-réseau et l'adresse IP du destinataire. On va alors faire un ET logique (*AND*) entre l'adresse IP de l'émetteur et son masque de sous-réseau pour trouver son network ID. Ensuite, on va faire un ET logique entre l'adresse IP du destinataire et le masque de sous-réseau de l'émetteur et comparer le résultat avec le network ID obtenu précédemment. Si les deux valeurs sont identiques, alors l'émetteur et le destinataire sont dans le même sous-réseau. Sinon, ils sont dans des sous-réseaux différents. Pas de panique, un exemple vaut mieux que tout ce pavé. 

### III.4.3.2. Le ANDing par l'exemple

Paul veut communiquer avec Éric. L'ordinateur de Paul (Paul-PC) a pour adresse IP 142.20.1.15 et pour masque de sous-réseau 255.255.0.0. Celui d'Éric (Éric-PC) a pour adresse IP 92.40.1.14.

#### III.4.3.2.1. Étape 1: déterminons le network ID de l'émetteur

Convertissons l'adresse IP de Paul-PC en binaire, ce que vous savez normalement faire. 🍌 Voici ce que vous devez obtenir :

10001110.00010100.00000001.00001111

Convertissons à présent le masque de sous-réseau de l'adresse IP de Paul-PC en binaire :

11111111.11111111.00000000.00000000

Nous allons à présent faire un AND entre ces deux groupes de nombres binaires en appliquant les règles précédentes :

$$\begin{array}{r} 10001110 . 00010100 . 00000001 . 00001111 \\ \text{AND } 11111111 . 11111111 . 00000000 . 00000000 \\ \hline = 10001110 . 00010100 . 00000000 . 00000000 \end{array}$$

FIGURE III.4.2. – Opération AND 1

Le network ID de Paul-PC est donc 10001110.00010100.00000000.00000000.

#### III.4.3.2.2. Étape 2: AND entre l'adresse IP du destinataire et le masque de sous-réseau de l'émetteur

Convertissons l'adresse IP d'Éric-PC en binaire :

01011100.00101000.00000001.00001110

Le masque de sous-réseau de Paul-PC ayant déjà été converti en binaire, il ne nous reste plus qu'à faire un ET logique :

$$\begin{array}{r} 01011100 . 00101000 . 00000001 . 00001110 \\ \text{AND } 11111111 . 11111111 . 00000000 . 00000000 \\ \hline = 01011100 . 00101000 . 00000000 . 00000000 \end{array}$$

FIGURE III.4.3. – Opération AND 2

On obtient donc 01011100.00101000.00000000.00000000.

### III. Veuillez vous identifier pour communiquer

#### III.4.3.2.3. Étape finale : comparaison des résultats

Au cours des deux étapes précédentes, nous avons obtenu :

É-	10001110.00010100.00000000.00000000
teur	
Des-	
ti-	01011100.00101000.00000000.00000000
na-	
taire	

Nous n'obtenons pas les mêmes valeurs. Par conséquent, ces deux adresses IP (142.20.1.15 et 92.40.1.14) ne sont pas dans le même sous-réseau.

Toujours à titre d'exemple, nous allons cette fois-ci choisir l'adresse IP du destinataire dans le même sous-réseau que celle de l'émetteur pour prouver que cette technique fonctionne bel et bien.

Si l'adresse IP d'Éric-PC est 142.20.20.10, sa notation en binaire sera 10001110.00010100.00010100.00001010

Nous avons déjà converti le masque de sous-réseau de l'émetteur en binaire donc nous pouvons passer directement au ET logique :

$$\begin{array}{l} \text{10001110 . 00010100 . 00010100 . 00001010} \\ \text{AND 11111111 . 11111111 . 00000000 . 00000000} \\ \hline = \text{10001110 . 00010100 . 00000000 . 00000000} \end{array}$$

FIGURE III.4.4. – Opération AND 3

Faisons une comparaison entre ce résultat et celui obtenu à l'étape 1 :

É-	10001110.00010100.00000000.00000000
teur	
Des-	
ti-	10001110.00010100.00000000.00000000
na-	
taire	

Comme vous pouvez le constater, le résultat est bien le même, donc ces deux adresses IP (142.20.1.15 et 142.20.20.10) sont dans le même sous-réseau. 🍊

Voilà, vous comprenez maintenant comment ça se passe. Chaque fois que vous communiquez, ce calcul logique est effectué. Si le destinataire est dans le même sous-réseau, l'hôte lui transmet directement les paquets, sinon, il les envoie au routeur (la passerelle) qui se charge de les **router au destinataire**.

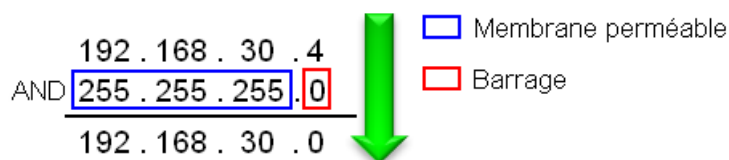
#### III.4.3.2.4. Un raccourci

La conversion du masque et de l'adresse IP en binaire n'est pas obligatoire, nous vous l'avons fait faire exprès pour vous faire pratiquer. 🍊 En fait, il est tout à fait possible de faire un ET logique directement en décimal. L'astuce c'est de se servir d'une analogie : une adresse IP représente de l'eau qui coule en direction du masque de sous-réseau. Chaque 255 du masque représente une membrane perméable qui laisse couler l'eau et chaque 0 représente un barrage qui bloque l'eau. Ainsi seuls les octets d'une adresse IP au-dessus d'un 255 se retrouveront dans le résultat du *ANDing*.



Ce raccourci ne marche que si on utilise les masques de sous-réseaux par défaut, c'est-à-dire composé uniquement de 255 et de 0 (ce qui est le cas dans notre exemple). Mais, si vous devez faire un ET logique qui implique des masques de sous-réseau personnalisés, souvent représentés par une notation CIDR, cette technique ne marchera pas. 🍊

Un schéma étant plus parlant qu'un discours, voici une illustration d'un ET logique entre l'adresse 192.168.30.4 et le masque 255.255.255.0 en utilisant le raccourci évoqué :



Les réseaux de zéro - zestedesavoir.com

FIGURE III.4.5. – Les bits allumés du masque forment une membrane perméable, les bits éteints, un barrage

Remarquez que chaque octet de l'adresse IP au-dessus d'un 255 « coule », c'est-à-dire se retrouve dans le résultat de la conjonction logique. Par contre, l'octet qui se trouve au-dessus d'un zéro est bloqué. 🍊

## Conclusion

Dans ce chapitre, nous avons fait un zoom sur la passerelle : nous avons étudié comment elle fonctionne, vu à quoi ça sert, et nous avons même étudié les opérations logiques qui s'effectuent au niveau de votre carte réseau. Tout ceci va vous servir de base pour l'étude de la fonction principale de la passerelle : le routage, que nous étudierons lorsque nous arriverons à la couche 3 du modèle OSI. Mais pour l'instant, nous sommes toujours coincés devant la couche 4 ! Courage, il ne nous reste plus que l'adressage par classes et l'adressage CIDR à voir... 🍊

## III.5. L'adressage par classes (obsolète)

### Introduction

Il y a fort fort longtemps, ~~dans l'antiquité~~ l'adressage se faisait **par classes**. En gros, pour chaque adresse IP, un masque de sous-réseau était assigné par défaut en fonction de sa classe. Cela ne se fait plus depuis bien des années, néanmoins, il peut être intéressant de voir ce que c'était pour mieux comprendre certaines notions. Nous allons donc voir quelque chose d'**obsolète**.

?

Ça sert à quoi de voir ça si ça date de Matusalem ? 🍌

Retournez voir votre professeur d'histoire que vous aviez au lycée ou au collège et posez-lui la question. 🍌 Vous voyez où on veut en venir ? Savoir le passé permet de comprendre le présent et d'appréhender l'avenir ! Allez hop, on y va ! (Le chapitre d'après portera sur l'adressage utilisé actuellement, ne vous inquiétez pas.)

### III.5.1. C'est quoi une classe ?

Au début de la deuxième partie, nous avons vu rapidement la notion d'adresse IP, sans rentrer dans les détails. Nous l'avons décrite comme un numéro d'une maison faisant partie d'une rue, ou encore comme un numéro de téléphone. Cette définition n'est pas très précise mais c'était le meilleur moyen de vous faire comprendre le principe. Dans ce chapitre, nous allons pousser notre investigation un peu plus loin à propos des adresses IPv4.

?

Hum, avant de commencer, pourriez-vous me montrer quelle est mon adresse IP ?

Pour voir votre adresse IP sous Windows, utilisez le raccourci clavier **[Windows] + [R]**, tapez **cmd** et validez. Sous Linux ou Mac OS, ouvrez votre terminal. Vous allez voir une belle boîte noire style DOS apparaître :

### III. Veuillez vous identifier pour communiquer

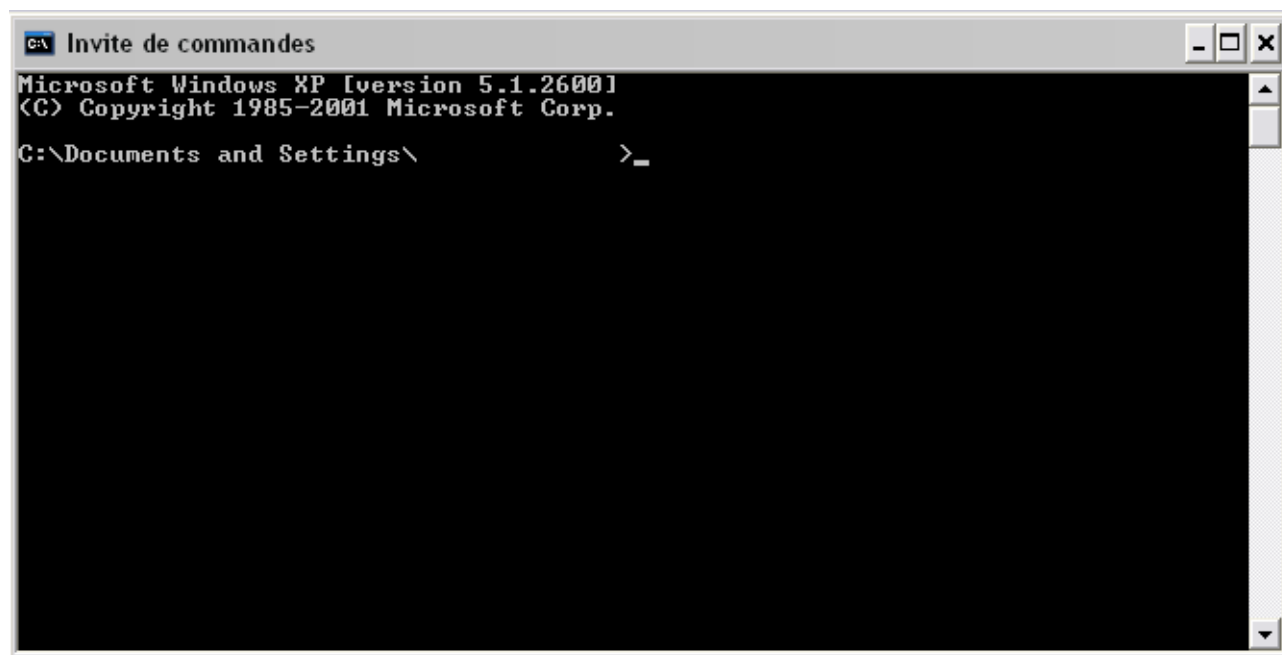


FIGURE III.5.1. – Invite de commandes sous Windows

Elle sera votre meilleure amie dans le monde du réseau, et nous allons beaucoup l'utiliser dans la suite du cours.

Ça fait peur hein ? 🍊 Il n'y a pas de quoi pourtant, même si ça paraît vieux, la console reste utile. Vous en aurez presque toujours besoin pour détecter les problèmes de réseau, vous vous en rendrez compte en temps voulu.

Dans l'invite de commandes, tapez `ipconfig` sous Windows ou `ifconfig` sous Linux et Mac OS. Vous obtenez alors un résultat tel que celui-là :

1	Configuration IP de Windows
2	
3	
4	Carte Ethernet Connexion au réseau local:
5	
6	Statut du média . . . . . : Média déconnecté
7	
8	Carte Ethernet Connexion réseau sans fil:
9	
10	Suffixe DNS propre à la connexion :
11	Adresse IP. . . . . : 192.168.1.17
12	Masque de sous-réseau . . . . . : 255.255.255.0
13	Adresse IP. . . . . :
	fe80::218:deff:fe39:75c%5
14	Passerelle par défaut . . . . . : 192.168.1.1

Vous n'avez peut-être pas les mêmes rubriques : cela dépend de votre type de connexion (Wi-Fi, filaire...). Repérez la rubrique où vous avez des informations sur l'IP, le masque de sous-réseau, etc. Il est possible que, comme dans le résultat ci-dessus, vous ayez deux adresses IP. Si c'est

### III. Veuillez vous identifier pour communiquer

le cas, cela veut dire que votre système d'exploitation supporte le protocole IPv6 : vous avez donc une adresse IPv4 et une IPv6. D'après les descriptions que nous vous avons donné dans la partie 1, vous devriez les reconnaître. 🍊

Bien évidemment, vous verrez probablement une ou des adresse(s) IP différente(s), avec peut-être un autre masque de sous-réseau. Vous pourrez aussi avoir une passerelle par défaut différente.

Voilà, vous savez donc à quoi ressemble votre adresse IP. 🍊

Mais, vous n'avez pas tous la même, c'est un fait. Alors, comme nous ne sommes pas des ségrégationnistes en matière d'adresses IP, nous allons nous occuper de toutes les classes.

?

Toutes les classes ? En réseau, les adresses sont *fashion* ? 🍊

Eh non, c'est bien un cours de réseau informatique, pas de mode. 🍊 Une classe en réseau est, en fait, un ensemble d'adresses IP. Chaque adresse IP appartient à une classe principale (on dit aussi *une plage*). Chaque classe a un masque de sous-réseau **par défaut**. Que vous le vouliez ou non, dès que vous donnez à votre carte réseau une adresse IP, votre système d'exploitation lui assigne directement un masque de sous-réseau par défaut selon la classe à laquelle appartient votre adresse IP.

Par convention, les créateurs du protocole IP disent qu'il existe 5 classes d'adresses IP. En d'autres termes, on peut choisir notre adresse IP dans ces cinq classes (théoriquement hein, parce que la pratique, c'est encore autre chose 🍊 ). Ces classes sont : A, B, C, D et E.

!

Rappel : actuellement, les classes sont obsolètes. On peut considérer qu'elles n'existent plus. Néanmoins, il peut être utile de voir de quoi il s'agissait. Gardez à l'esprit, lors de la lecture de la suite de ce chapitre, cette information. Comment ça, on radote ? 🍊

Maintenant que les présentations sont faites, étudions-les un peu plus en détail !

## III.5.2. Classe A

Commençons par l'étude de la classe A.

### III.5.2.1. Présentation

Nous vous avons dit qu'une classe d'adresses IP est en fait un ensemble d'adresses. Dans le cas de la classe A, ces adresses IP se situent entre 1.0.0.0 et 127.255.255.255. Son masque de sous-réseau par défaut est 255.0.0.0. En pratique, les adresses IP de la classe A se situent entre 1.0.0.0 (compris) et 126.255.255.255.

?

Mais alors, à quoi servent les adresses IP entre 127.0.0.0 et 127.255.255.255 ?

### III. Veuillez vous identifier pour communiquer

En fait, les adresses IP commençant par 127 sont utilisées pour faire des tests particuliers. Faisons un test. Reprenez votre invite de commandes, ou terminal (on vous l'avait bien dit que vous alliez beaucoup l'utiliser 🍊). Sous Windows, tapez :

```
ping 127.0.0.1
```

Ou sous Linux :

```
ping -c 4 127.0.0.1
```

i

On verra plus tard ce qu'est ping en détails. Pour faire court : c'est un outil de diagnostic.

Si le protocole TCP/IP est correctement implémenté, c'est-à-dire si votre système d'exploitation est capable de se connecter à un réseau (on peut supposer que c'est le cas vu que vous êtes en train de lire cette page 🍊), vous aurez une suite de réponses de votre carte réseau, généralement 4 lignes. Nous vous laissons lire et comprendre ces quelques lignes, vous en êtes largement capables. 🍊

Revenons à l'étude de l'adresse 127.0.0.1. On l'appelle **loopback address**. D'accord, c'est de l'anglais... Cependant, dans le monde du réseau, c'est la langue principale, c'est donc important d'apprendre ce vocabulaire. 🍊

On va traduire ça ensemble. Le mot *loopback* signifie « boucle de retour ». Donc, lorsque vous faites `ping 127.0.0.1`, vous faites en réalité un ping vers... votre ordinateur ! En fait, votre système d'exploitation crée automatiquement un réseau spécial composé uniquement de lui-même. Sous Linux, ce réseau spécial est représenté par l'interface **lo**.

?

Quel intérêt ?

Eh bien, cela permet de tester des applications réseau sans être connecté réellement à un réseau. Par exemple, vous voulez tester un script PHP (ce qui nécessite un logiciel serveur Apache, généralement) mais vous n'êtes pas connecté à Internet, vous ne pouvez pas l'envoyer sur un serveur pour le tester. Votre ordinateur peut faire office de serveur, grâce à WAMP ou un logiciel de ce genre. Pour tester votre script, votre ordinateur se connecte à lui-même et s'envoie lui-même des requêtes. Ça paraît tordu comme ça, mais en fait c'est logique. 🍊

i

Notez que, sous Windows et Linux, toute adresse de la forme 127.XXX.XXX.XXX marchera à la place de 127.0.0.1. Si vous voulez, vous pouvez tester avec ping.

À travers cet exemple (certes un peu long), vous voyez qu'on ne peut pas utiliser les adresses 127.XXX.XXX.XXX.

Revenons maintenant à l'étude de la classe A. Ses adresses IP sont généralement utilisées dans de très très grandes entreprises et chez les FAI. Vous vous demandez pourquoi ? Pour répondre, il va falloir nous intéresser à la structure d'une adresse IP.

### III.5.2.2. Structure d'une adresse IP de la classe A

Prenons une adresse IP de la classe A. Au hasard : 110.0.0.1. Si vous avez bien retenu ce que nous avons dit plus haut, son masque de sous-réseau par défaut est 255.0.0.0. 🍊

Schématiquement, ça donne ceci :

110 . 0 . 0 . 1  
└─  
1 octet = 1 byte = 8 bits

Les réseaux de zéro - zestedesavoir.com

FIGURE III.5.2. – Décomposition d'une adresse IP de classe A

Une adresse IP est constituée de 4 octets. Votre ordinateur, lui, ne « voit » pas une adresse IP, comme vous et nous. Nous voyons des nombres décimaux tandis qu'il « voit » des nombres binaires, une suite de 0 et de 1 (à supposer que les ordinateurs « voient » 🍊).

Dans notre exemple, c'est-à-dire dans le cas d'une adresse IP de la classe A, le premier octet est **l'identité du réseau**, soit en anglais **network ID**.

?

Qu'est-ce que c'est ?

Cela indique simplement que **l'adresse client 0.0.1 se trouve dans le réseau 110**. Donc, à ce niveau, vous avez dû comprendre que la partie **0.0.1** est l'adresse de votre carte réseau. On l'appelle **l'adresse client**, ou, en anglais, **host ID**.

i

Si vous avez une adresse IP de 110.0.0.1, vous pouvez communiquer avec tous les hôtes de votre réseau (qui auront donc pour adresse IP 110.XXX.XXX.XXX). Par contre, si vous voulez communiquer avec un hôte dans le réseau 122, il vous faudra passer par... **une passerelle (un routeur)**. 🍊 Vous ne l'aviez pas oublié, si ?

!

Notons une règle d'or : dans un réseau, deux clients (ordinateurs, imprimantes, etc.) ne peuvent pas utiliser une même adresse IP, de même que, dans un pays, 2 lignes téléphoniques ne peuvent pas avoir le même numéro attribué. 🍊

Bref, cela explique pourquoi ce sont les très grandes entreprises et les FAI qui utilisent ce type d'adresses. En effet, grâce à la répartition des octets entre network ID et host ID, vous pouvez avoir 16 777 214 adresses IP **par** réseau. De plus, vous pouvez avoir un total de 126 réseaux. Vous comprenez donc que ça intéresse les FAI qui doivent donner des adresses IP à un très grand nombre de personnes. 🍊



Un sous-réseau en anglais se dit **subnet** qui est le diminutif de *subnetwork*.

A priori, vous ou nous n'aurons pas vraiment affaire à cette classe, même dans le monde professionnel sauf si, bien sûr, vous travaillez pour des FAI. Dans ce cas, des formations telles que CISCO CCNA, CCNP, voire CCIE vous seront utiles. 🍊

### III.5.3. Classes B et C

À présent, intéressons-nous aux classes B et C.

#### III.5.3.1. Classe B

Premièrement, parlons de la classe B. Il n'y a pas grand-chose à dire, voilà pourquoi elle vient en premier (et aussi parce que c'est l'ordre alphabétique 🍊).

##### III.5.3.1.1. Présentation

Les adresses IP de la classe B sont celles entre 128.0.0.0 et 191.255.255.255. Le masque de sous-réseau par défaut de cette classe est 255.255.0.0.

Seules des grandes ou moyennes entreprises vont utiliser ce type d'adresses IP pour raccorder plusieurs ordinateurs car dans la classe B, on a une possibilité de 65 534 ordinateurs **par** réseau. Comme pour la classe A, ce nombre vient de la structure des adresses IP de la classe B que nous allons étudier maintenant plus en détails. 🍊

##### III.5.3.1.2. Zoom sur la structure d'une adresse IP de la classe B

Prenons une adresse de la classe B pour notre étude. Par exemple : 172.40.0.5 (en fait, vous n'avez pas vraiment le choix 🍊).

La partie **172.40** est l'identité réseau et la partie **0.5** est l'identité client. On dit que l'adresse 0.5 se trouve dans le réseau 172.40. C'est le même principe que pour la classe A. 🍊



Mais pourquoi l'identité réseau prend deux octets dans ce cas ?

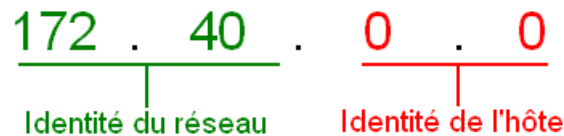
C'est déjà bien si vous vous êtes posé cette question. Sinon, relisez ce chapitre : vous devez absolument bien comprendre les notions de *bits*, octets et *bytes*.

Bref, c'est grâce à une question comme celle-là que l'on se rend compte de l'importance d'un masque de sous-réseau. En effet, celui-ci **définit** quelles parties des 4 de votre adresse IP (ou quels octets de votre adresse IP) correspondent à l'identité réseau et quelles parties correspondent à l'identité client.

### III. Veuillez vous identifier pour communiquer

Quand on était dans la classe A, on avait un masque de sous-réseau de 255.0.0.0. Or, dans une adresse IP de la classe A telle que 110.0.0.1, seul le premier octet (ici 110) correspond à l'identité réseau. Maintenant, regardez son masque de sous-réseau, seul le premier octet est à 255, les autres sont à 0. Nous pensons que là vous avez compris comment ça fonctionne. 🍊

Reprenons notre adresse de la classe B. Comme dans notre masque de sous-réseau les deux premiers octets sont à 255, dans notre adresse IP, les deux premiers octets correspondent à l'identité réseau :



Les réseaux de zéro - zestedesavoir.com

FIGURE III.5.3. – Décomposition d'une adresse IP de classe B

#### III.5.3.2. Classe C

Abordons maintenant la classe C. Nous allons en parler un peu plus longtemps.

##### III.5.3.2.1. Présentation

Les adresses de la classe C sont entre 192.0.0.0 et 223.255.255.255. Le masque de sous-réseau par défaut est 255.255.255.0.

Cette classe est celle qui nous intéresse le plus. En effet, la plupart de nos adresses IP que nous avons vues en début de chapitre sont dans cette classe. 🍊 Après cela dépend aussi de votre FAI. Certains vous donneront des adresses privées et utiliseront des services comme NAT pour vous donner accès à Internet. Vous aurez plus d'informations sur les classes privées dans une autre sous-partie. 🍊

Dans cette classe on peut avoir 254 adresses IP par réseau, et 2 097 152 réseaux.

?

Pourquoi seulement 254 adresses IP par réseau ? De 1 à 255, ça en fait 255, non ?

Bonne remarque. 🍊 Pour répondre, il va nous falloir faire un passage rapide et indolore sur...

##### III.5.3.2.2. Les envois de données

Dans un réseau informatique, il y a plusieurs moyens d'envoyer des données.

- **L'unicast** : dans ce cas, on envoie des données à un seul ordinateur ;
- **Le multicast** : l'envoi des données se fait vers un groupe d'ordinateurs ;
- **Le broadcast** : on envoie des données à tous les ordinateurs du réseau.

### III. Veuillez vous identifier pour communiquer

Ce qu'il faut savoir, c'est que l'adresse 255 dans les réseaux de la classe C est **une adresse de *broadcast* réseau**. Nous avons déjà vu cette notion : si vous envoyez des données à cette adresse, les données seront envoyées à tous les ordinateurs du réseau. On a la même chose pour les adresses des classes A et B. Par exemple, l'adresse 255.255.255 du réseau 110 (pour la classe A) est une adresse de *broadcast*, ainsi que l'adresse 255.255 du réseau 140.20 (pour la classe B).

Un hôte ne peut donc pas prendre cette adresse IP, puisqu'elle sert, en quelque sorte, de support à l'envoi de données, ce qui explique qu'on ait seulement 254 adresses IP par réseau.

#### III.5.3.2.3. Structure d'une adresse IP de la classe C

Bon, je pense que vous avez compris le principe. 🍏 Pour vérifier ça, vous allez faire un exercice : prenez une adresse IP de la classe C au hasard, écrivez son masque de sous-réseau et dites quelle partie correspond à l'identité réseau et quelle partie correspond à l'identité client. Honnêtement, si vous n'êtes pas capables de faire cet exercice, nous insistons, **relisez tout le chapitre depuis le début !**

Voici une correction :

👁 Contenu masqué n°3

Passons maintenant aux dernières classes qui nous restent à étudier.

#### III.5.4. Classes D et E

Consacrons-nous à présent aux classes D et E qui sont, en fait, sans grand intérêt pour nous. 🍏

En effet, vous pouvez vous permettre d'ignorer ces deux classes. Mais bon, pour le principe, nous allons quand même vous expliquer pourquoi on choisit de les ignorer et vous donner quelques informations.

?

Pourquoi ignorer les classes D et E ?

Dans la pratique, vous n'utiliserez pas la classe D. Quant à la classe E, vous ne pouvez pas l'utiliser : c'est une classe **expérimentale**. Seuls les développeurs du protocole TCP/IP l'utilisent pour des expériences. 🍏

##### III.5.4.1. Quelques informations...

Pour la culture, nous allons vous donner quelques informations. Pour les classes D et E, le nombre de réseaux possibles et le nombre d'identités client ne sont pas connus : c'est **non-assigné**. On pourrait supposer qu'on utilisera des adresses de ces deux classes dans les années à venir mais,

### III. Veuillez vous identifier pour communiquer

en réalité, cela ne risque pas d'arriver. En effet, il y a déjà une nouvelle version du protocole IP : IPv6. Il est donc logique que l'on migre vers le protocole IPv6.

Voici les portées de ces deux classes :

- Classe D : de 224.0.0.0 à 239.255.255.255 ;
- Classe E : de 240.0.0.0 jusqu'à 255.255.255.255.

Vous pouvez vous amuser à compter le nombre d'adresses IP possibles dans ces deux classes, cependant officiellement cela n'a pas été assigné. 🍊

### III.5.5. Notion de classe privée

Nous allons à présent aborder la notion de classe privée.

?

Qu'est-ce qu'une classe privée ?

Une classe privée est une portée d'adresses IP d'une certaine classe publique (A, B, C), mais réservée pour un usage particulier par des standards ou conventions. 🍊

Par exemple, la portée 169.254.0.0 - 169.254.255.255 fait partie d'une classe privée réservée par Microsoft pour le protocole APIPA. Celui-ci permet la configuration automatique des adresses IP lorsque aucune autre configuration n'est apportée. Sous Windows, quand aucune adresse IP n'est explicitement définie, le protocole DHCP tente de contacter d'autres éléments sur le réseau afin d'en obtenir une. Si aucune réponse n'est obtenue, c'est alors qu'on se retrouve avec une adresse de cette plage.

Microsoft a réservé cette classe pour l'assignation automatique des adresses IP lorsque aucune autre configuration manuelle ou dynamique n'a été faite. Cette portée d'adresse (ou classe privée) est très importante dans le diagnostic des ordinateurs qui tournent sous Windows.

Il y a aussi la portée des adresses IP de 192.168.0.0 - 192.168.255.255. Beaucoup d'entre vous ont une adresse IP commençant par 192.168, nous y compris. Comme ces adresses IP sont issues d'une classe privée, il faut donc utiliser un service particulier pour pouvoir accéder à Internet : ce service, c'est PAT que nous verrons plus tard. 🍊

Si vous avez une adresse IP d'une classe publique, vous n'avez éventuellement pas besoin d'une passerelle car vous avez accès au réseau public qu'est Internet. Mais un FAI « sérieux » vous donnera une adresse IP privée même si elle est dans la classe A, B, ou C. Oui, il y a aussi des classes privées dans les classes A et B. Dans la classe A les adresses IP allant de 10.0.0.0 à 10.255.255.255 sont des adresses privées, et dans la classe B, celles allant de 172.16.0.0 à 172.31.255.255 sont des adresses privées aussi.

## Conclusion

Maintenant que vous connaissez cela, revenons dans notre époque et intéressons-nous à l'adressage CIDR, beaucoup plus actuel !

*III. Veuillez vous identifier pour communiquer*

## Contenu masqué

### Contenu masqué n°3

- Adresse IP : 194.220.34.1
- Masque de sous-réseau par défaut : 255.255.255.0
- Identité réseau : 194.220.34
- Identité client : 1

[Retourner au texte.](#)

## III.6. L'adressage CIDR

### Introduction

Nous vous avons dit en parlant des classes que ces dernières avaient presque entièrement disparu parce qu'elles étaient devenues obsolètes. Elles ont été remplacées par un système d'adressage plus fiable et plus performant, à savoir l'adressage CIDR, qui est l'objet de ce chapitre. Étant donné qu'il s'agit du système d'adressage utilisé actuellement, vous feriez mieux d'être plus concentrés que lorsque nous avons traité des classes. 🍊 C'est parti !

#### III.6.1. Révision de l'adressage par classes

Pour ne pas transformer cette section en cours d'histoire, nous vous proposons de lire [cet article](#) ↗ pour plus d'informations à caractère « culturel » sur la naissance et l'évolution d'Internet.

Nous allons revoir le fonctionnement de l'adressage par classes pour vous préparer à la suite du tutoriel.



Les cours sur les réseaux ne sont pas toujours à jour, aussi la majorité des étudiants continue à croire que les classes sont toujours d'actualité. Elles ont bien existé, mais elles ont été supplantées par l'adressage CIDR.

L'adressage par classes est un système utilisant une architecture réseau appelée en anglais *classful network*, c'est-à-dire « réseau dans les classes ». Cette expression traduit que le principe — majeur, sinon l'unique — de ce système d'adressage est de répartir les adresses IP par classes.

Le réseau ayant pour but de permettre la communication entre machines, les créateurs de la pile TCP/IP se sont inspirés du monde réel pour créer un système de communication informatique. Dans une société donnée, des individus vivent dans des maisons (en général), parlent une langue, ont un nom unique, vivent parfois dans des villes, occupent certaines fonctions, ont des responsabilités et des droits. Un système de communication informatique reprend plus ou moins ces grandes lignes : des individus (hôtes) parlent des langues (protocoles), ont des noms uniques (adresses IP), vivent dans des maisons (sous-réseau ou réseau, c'est selon), occupent des fonctions (clients, serveurs, passerelles, routeurs, etc.), ont des responsabilités (transmettre des données à la demande du client, distribuer automatiquement des adresses IP pour le cas d'un serveur DHCP) et des droits (réclamer un renouvellement d'adresse IP, demander l'identité d'un autre hôte, exiger un mot de passe, etc.).

### III. Veuillez vous identifier pour communiquer

Ainsi, pour rendre opérationnel ce tout nouveau système de communication, il y avait une espèce de sac plein de noms (adresses IP). Alors, on assignait un nom (une adresse IP) à chaque individu (hôte). Pour gérer cette distribution d'adresses, on a créé l'adressage par classes.

Pour reprendre un exemple, imaginez que vous ayez 10 000 prénoms à attribuer à autant de nouveau-nés. Pour mieux gérer cette attribution, vous pourriez classer les prénoms par ordre alphabétique : tous les prénoms commençant par un A sont réunis dans un dossier « Prénoms A », etc. Alors, si une dame venait vous demander dix prénoms pour ses dix futurs enfants, vous n'auriez qu'à lui demander les contraintes auxquelles doivent répondre ces prénoms :

« Je veux des prénoms qui commencent par la lettre A et constitués de cinq lettres », dirait-elle.

Rien de plus simple que de regarder votre dossier « Prénoms A » et d'en choisir dix. Mais que se passerait-il si, après avoir reçu les dix prénoms, la dame ne mettait au monde qu'un seul enfant (exemple très original 🍌) ? Quid des neuf autres prénoms ? Ce serait du gaspillage !

Le système d'adressage par classes fonctionne selon le même principe : les adresses IP sont rangées par classes et dans chacune d'elles se trouvent des plages. Si une entreprise demandait des adresses pour cent ordinateurs, on choisirait la classe lui offrant ce nombre d'adresses et on lui offrirait des adresses IP issues de cette classe.

Le problème de ce système d'adressage est le pourcentage assez élevé de perte d'adresses. Nous avons vu que toutes les adresses IP de la classe A, par exemple, nous permettaient d'obtenir 16 777 214 adresses IP par réseau en utilisant les masques par défaut. Cela dit, l'entreprise qui voudrait une adresse IP pour un réseau de 10 000 hôtes aurait quand même 16 767 214 d'adresses en surplus. Quelle perte !

Si l'adressage par classes n'avait pas été remplacé depuis les années 1990, aujourd'hui nous utiliserions presque exclusivement les adresses IPv6, car nous aurions très vite connu une pénurie d'adresses. C'est pourquoi un nouveau système d'adressage, capable de réduire au minimum le gaspillage d'adresses IP et de faciliter considérablement le routage, a été mis en place. Nous allons le voir dans la prochaine section.

#### III.6.2. CIDR et le supernetting



La compréhension de cette section demande beaucoup de concentration. Vous êtes priés de rester attentifs tout au long de votre lecture. Prenez une feuille et un crayon pour faire les calculs au même moment que nous. Si vous vous contentez de les lire comme on lirait un roman, vous les trouverez très difficiles. Nous ne nous attendons pas non plus à ce que vous maîtrisiez le *supernetting* du premier coup : même des étudiants ont du mal à appréhender cette notion.

L'adressage sans classes (ou adressage CIDR) est le système de gestion et d'allocation d'adresses IP le plus utilisé aujourd'hui. Ce système, qui est régi par les [RFC 1518](#) et [1519](#), a été conçu pour remplacer l'adressage par classes pour les raisons que nous avons évoquées dans les chapitres précédents. Le but de ce nouveau système s'articule autour de deux points :

### III. Veuillez vous identifier pour communiquer

- Économiser les adresses IP.
- Faciliter le routage.

?

CIDR ? Ce ne serait pas plutôt cidre ? 🍷

Nous parlons de réseaux et non de boissons... 🍷 CIDR est l'acronyme de *Classless Inter Domain Routing* (« routage sans classes entre domaines »). Plutôt bizarre, non ? En bref, par CIDR comprenez « routage effectué entre domaines qui n'utilisent pas les classes ». On comprend alors que le réseau Internet est fondé sur ce système d'adressage. Logique, quand on y pense... Sinon, comment un système d'adressage par classes aurait-il pu supporter plus de 2 milliards d'internautes ? Depuis les années quatre-vingt-dix, nous n'aurions plus d'adresses IP disponibles.

i

En anglais, les adresses IP utilisant l'adressage CIDR sont appelées *classless addresses* par opposition aux *classful addresses*, qui désignent celles qui utilisent l'adressage par classes. Habituez-vous à ce vocabulaire qui est très présent dans les documentations en anglais.

Quand nous parlons d'assignation d'adresses IP, en tant qu'administrateur d'un réseau, nous devons examiner deux choses :

- Les contraintes administratives pour obtenir et allouer les adresses.
- L'aspect technique (sous-entendu le routage, le plus souvent) que cela implique.

CIDR répond mieux aux contraintes techniques.

#### III.6.2.1. CIDR: le comment

Nous allons maintenant nous focaliser sur le comment, étant donné que vous savez déjà pourquoi ce nouveau système a été créé.

Soit l'adresse 192.168.10.0/23. À ce stade, vous êtes censés savoir que le nombre après le slash (/) équivaut au nombre de bits masqués. Si vous avez encore des difficultés, nous vous recommandons la relecture de la sous-partie sur la notation du masque.

Bien ! 192.168.10.0/23 applique un masque de 255.255.254.0 au réseau 192.168.10.0. Grâce à cette notation, nous pouvons calculer (et vous êtes censés savoir le faire seuls à présent) l'étendue du sous-réseau qui ira donc de 192.168.10.0 à 192.168.11.255 (si nous incluons l'adresse de diffusion ou *broadcast address*), dans un réseau sans classes. Par contre, si nous étions dans un réseau n'utilisant pas l'adressage CIDR, **192.168.10.0/23** représenterait **une fusion de deux sous-réseaux de la classe C, 192.168.10.0 et 192.168.11.0 ayant chacun un masque de sous-réseau de... 255.255.255.0.**

Cela dit, avec l'adressage CIDR, le masque /23 nous donne l'équation suivante :

192.168.10.0/23 (adressage CIDR) = 192.168.10.0/24 (ou 255.255.255.0) + 192.168.11.0/24 (ou 255.255.255.0)

### III. Veuillez vous identifier pour communiquer

Vous voyez ? Nous avons la possibilité d'utiliser un seul réseau qui fusionne plusieurs sous-réseaux. Cette fusion de sous-réseaux, dite aussi *supernetting*, est l'essence même de CIDR. Cette technique est également appelée résumé de routes (*route summarization* en anglais).

i

Cette pratique viole la règle d'or du *subnetting*. Vous vous en souvenez ? Celle des 0 (*network ID*) et des 1 (*broadcast address*). Tous les routeurs qui supportent ce type d'adressage ignorent également cette règle.

Pour implémenter un réseau fondé sur l'adressage CIDR, il faut utiliser un protocole qui puisse le supporter. Il en existe plusieurs, tels que BGP et OSPF. Si le protocole ne supporte pas ce type d'adressage, le routage échouera dans ce réseau. En général, les petits LAN et les réseaux « maison » n'implémentent pas l'adressage CIDR.

#### III.6.2.2. Comment résumer une route

Dans l'adressage par classes, nous utilisons le *subnetting* pour réduire la congestion d'un réseau en le subdivisant en plusieurs sous-réseaux. Toujours est-il que nous perdions quelques adresses IP, étant donné que plusieurs sous-réseaux utilisaient un même masque. Cela dit, chaque sous-réseau avait le même nombre d'adresses. « Supernetter » un réseau est exactement le contraire de « subnetter » un réseau, sauf qu'ici, il ne s'agit plus de l'adressage par classes mais de l'adressage CIDR. Tous ces sous-réseaux peuvent donc être fusionnés et rassemblés sous un seul préfixe.

Un exemple vaut mieux que tout ce pavé. 🍊

Si nous avons quatre *subnets* tels que :

- *Subnet 1* : 192.168.0.0/24 soit 11000000. 10101000. 00000000.00000000/24
- *Subnet 2* : 192.168.1.0/24 soit 11000000. 10101000. 00000001.00000000/24
- *Subnet 3* : 192.168.2.0/24 soit 11000000. 10101000. 00000010.00000000/24
- *Subnet 4* : 192.168.3.0/24 soit 11000000. 10101000. 00000011.00000000/24

Nous remarquons que ces quatre *subnets* ont bien le même préfixe /24 : nous pouvons les fusionner sous un seul préfixe. Par conséquent, nous obtenons la route suivante : **192.168.0.0/22 soit 11000000.10101000.00000000.00000000/22.**

?

Comment avez-vous obtenu le /22 alors que nous avions /24 au départ ?

Très belle question ! Nous avons simplement appliqué la technique d'agrégation de routes. Supernetter, c'est la même chose qu'agréger des routes. Le résultat obtenu est donc appelé route agrégée ou route résumée. Pour obtenir le /22, nous avons suivi quatre étapes bien précises que vous devez suivre également.

### III. Veuillez vous identifier pour communiquer

#### III.6.2.2.1. Étape 1: détecter les réseaux ayant le même préfixe

Dans cette étape, nous avons pris quatre réseaux ayant le même préfixe (/24) : il s'agit de 192.168.0.0, 192.168.1.0, 192.168.2.0 et 192.168.3.0.

#### III.6.2.2.2. Étape 2: convertir des réseaux en binaire

Ensuite, nous avons converti chaque adresse réseau en binaire. Pourquoi ? Parce que c'est important pour l'étape 3.

#### III.6.2.2.3. Étapes 3 et 4: détecter les motifs entre les sous-réseaux en binaire (étape 3) et les compter (étape 4)

Ne paniquez pas : ce n'est pas difficile. Quand nous avons converti les quatre sous-réseaux en binaire, qu'avons-nous obtenu ?

```
— Subnet 1 : 11000000. 10101000. 00000000. 00000000
— Subnet 2 : 11000000. 10101000. 00000001. 00000000
— Subnet 3 : 11000000. 10101000. 00000010. 00000000
— Subnet 4 : 11000000. 10101000. 00000011. 00000000
```

Y a-t-il quelque chose de commun à ces quatre sous-réseaux ? Rien ? Vous en êtes sûrs ? Nous allons vous faciliter la tâche.

```
— Subnet 1 : 11000000.10101000.00000000.00000000
— Subnet 2 : 11000000.10101000.00000001.00000000
— Subnet 3 : 11000000.10101000.00000010.00000000
— Subnet 4 : 11000000.10101000.00000011.00000000
```

Et maintenant ? 🍊

Tous ces sous-réseaux ont 11000000.10101000.000000 en commun. Comptons le nombre de bits : il y en a 22, n'est-ce pas ? 22 sera donc notre nouveau préfixe. Le *network ID* sera la plus petite adresse IP parmi les quatre, soit 192.168.0.0. Enfin, la nouvelle route, la route résumée ou agrégée, sera **192.168.0.0 /22**.

Voilà, vous savez tout sur le *supernetting* et le *subnetting*. Nous y reviendrons certainement une fois que vous maîtriserez le routage, afin que nous constations combien il est efficace de résumer les routes pour ne pas alourdir la table de routage.

#### III.6.2.3. Quelques exercices pour la route

Ne croyez pas que nous allons vous laisser filer comme ça, il vous faut pratiquer et encore pratiquer !

### III. Veuillez vous identifier pour communiquer

#### III.6.2.3.1. Exercice 1: supernetting

Votre premier exercice est relativement simple. Plus haut, nous avons pris le cas d'un réseau 192.168.10.0/23 et nous avons évoqué l'équation suivante :

192.168.10.0/23 (adressage CIDR) = 192.168.10.0/24 (ou 255.255.255.0) + 192.168.11.0/24 (ou 255.255.255.0)

Prouvez que **192.168.10.0/23** est bel et bien une fusion (une route agrégée) de **192.168.10.0/24** et **192.168.11.0/24**. C'est très simple, il suffit de respecter les étapes que nous avons définies plus haut.

#### III.6.2.3.2. Exercice 2: stagiaire chez Link it Technology

Vous êtes stagiaire dans une entreprise éditrice de logiciels nommée *Link it Technology*. Le réseau de ladite entreprise est constitué de 192 hôtes parmi lesquels 4 serveurs :

- **Srvprog** est le serveur que les programmeurs (au nombre de 47) de l'entreprise utilisent. Il héberge un nombre important d'applications.
- **Srvcomp** est le serveur des 76 comptables. Il héberge également un nombre important d'applications de comptabilité.
- **Srvprint** est le serveur d'impression des secrétaires. On en compte 33 qui effectuent un nombre considérable d'impressions par jour, ce qui alourdit le réseau et empêche aux autres services de communiquer plus rapidement avec leurs serveurs respectifs.
- **Srvboss\_backup** est le serveur sur lequel sont sauvegardés tous les fichiers des chefs de division. Le système de *backup* de l'entreprise est automatique : chaque fois qu'un fichier est modifié et sauvegardé, une copie est sauvegardée aussitôt sur ce serveur. Les chefs de division, tasse de café à la main chaque matin, modifient plusieurs fichiers. Ils sont au nombre de 36 (les chefs, pas les fichiers!).

Votre chef se plaint de la congestion du réseau et vous demande de mettre en place un plan d'adressage pour minimiser le trafic. Vous devrez donc, à partir de l'adresse réseau 120.12.0.0/18, aboutir à une solution satisfaisante. Dans ce cas précis, il vous est demandé de subnetter ce réseau en quatre :

- Un réseau **netprog** pour tous les développeurs de l'entreprise et leur serveur.
- Un réseau **netcomp** pour tous les comptables et leur serveur.
- Un réseau **netsecr** pour tous les secrétaires et leur serveur de fichiers.
- Un réseau **netbackup** pour tous les chefs de division et leur serveur *backup*.



Pour cet exercice, nous supposons que les sous-réseaux ne communiquent pas entre eux, donc que le trafic reste interne. Par conséquent, nous n'avons pas besoin de routeurs et de calculer les plages pour leurs interfaces. Dans la réalité, ce calcul est obligatoire mais ignorez cette étape pour cet exercice. Vous le ferez dans la prochaine sous-partie.

À partir de l'énoncé ci-dessus, votre mission est triple :

- Déterminer le network ID de chaque *subnet* et leur masque.
- Déterminer les plages d'adresses de chaque *subnet* en incluant leur *broadcast address*.

### III. Veuillez vous identifier pour communiquer

- Appliquer la technique du supernetting pour avoir une route résumée des *subnets* que vous aurez obtenus.

i

Les sous-réseaux n'ont pas le même nombre d'hôtes ; ainsi pour déterminer combien de bits il faut masquer, vous devrez vous focaliser sur le plus grand sous-réseau. Dans notre scénario, il s'agit du sous-réseau netcomp (76 hôtes pour les comptables). Trouvez un masque qui vous permette d'avoir au moins 76 hôtes par sous-réseau. Nous perdrons des adresses, certes, mais vous ne savez pas encore comment implémenter des masques à longueur variable. Pour cela, rendez-vous à la prochaine sous-partie.

👁 Contenu masqué n°4

### III.6.3. Les masques à longueurs variables (VLSM)

!

Comme précédemment, la compréhension de cette section demande beaucoup de concentration. Papier et crayon vous seront utiles pour faire des calculs et prendre des notes. Si vous vous contentez de lire, vous risquez de ne pas retenir grand-chose !

VLSM, pour *Variable Length Subnet Mask* (soit **masque de sous-réseaux à longueur variable**) est une technique utilisée dans le but de mieux gérer les adresses IP, tout comme le CIDR. En fait, VLSM est une extension de CIDR. La différence est que le CIDR est plus utilisé au niveau internet et le VLSM est plus utilisé dans un réseau local, mais les deux permettent de minimiser la perte d'adresses. 🍏

i

Pour mettre en place un réseau aux masques à longueurs variables, il faut être sûr que les routeurs supportent les protocoles compatibles au VLSM. Quelques-uns de ces protocoles sont **OSPF, EIGRP, RIPv2, IS-IS**. Vous n'avez pas besoin de connaître ce qu'ils sont et ce qu'ils font, nous étudierons quelques-uns d'entre eux en temps voulu. 🍏

#### III.6.3.1. Son utilité ?

Pour comprendre à quoi sert l'implémentation des masques de sous-réseaux variables, nous allons considérer un scénario.

Vous avez un réseau de 250 hôtes. Vous voulez réduire la congestion de ce dernier et décidez de le subnetter en plusieurs sous-réseaux. Grâce aux techniques du subnetting et aux règles que vous avez apprises, vous décidez de le subnetter en cinq réseaux de 50 hôtes chacun.



Il est impossible d'obtenir cinq réseaux de 50 hôtes pile chacun (d'ailleurs, c'est une mauvaise pratique de choisir un masque qui nous donne exactement le nombre d'hôtes dont nous avons besoin). À la rigueur, nous pourrions obtenir cinq réseaux pouvant contenir au moins 50 hôtes. Dans ce cas, le réseau pourrait contenir un maximum de 62 hôtes si on masque 2 bits. 🍊

Vous obtenez alors vos cinq sous-réseaux et vous êtes contents : le but est atteint. 🍊

Maintenant, imaginez que vous êtes un administrateur réseau employé dans une entreprise. Vous avez un sous-réseau **192.168.100.0/24**. Votre patron vous dit qu'il souhaite une segmentation par fonctions, comme nous l'avons étudié dans l'analyse des contraintes et plan d'adressage. Il vous donne les spécifications suivantes :

- Un sous-réseau de 50 hôtes, uniquement pour les secrétaires de l'entreprise.
- Deux sous-réseaux de 12 hôtes chacun, pour les techniciens et les comptables.
- Un sous-réseau de 27 hôtes pour les développeurs d'applications.

Pour une meilleure compréhension de ce qui nous est demandé, considérons le schéma ci-dessous auquel vous devez vous référer.

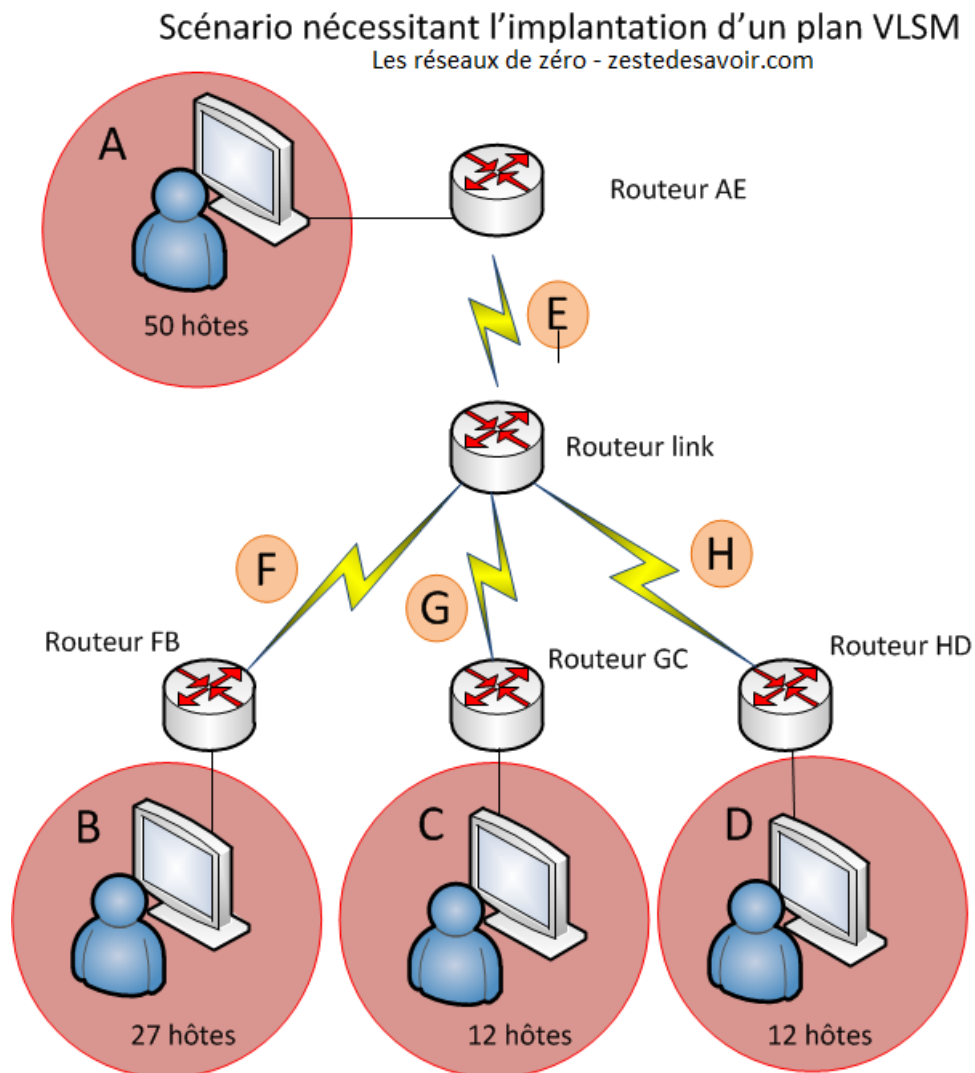


FIGURE III.6.1. – Scénario nécessitant l’implantation d’un plan VLSM

Les cercles rouges représentent les sous-réseaux que nous voulons obtenir. Nous avons au total cinq routeurs :

- *routeur\_AE*, qui relie le réseau A et E ;
- *routeur\_link*, qui relie les réseaux F, G et H au réseau E ;
- etc.



N.B. : les réseaux E, F, G et H sont en orange pour une raison précise. Il s’agit en fait des interfaces de liaison.

Comment allez-vous mettre cela en place en subnettant ? Ce n’est pas possible, car le subnetting nous permet d’avoir plusieurs sous-réseaux ayant un même nombre d’hôtes et un même masque, mais ayant des portées d’adresses différentes pour marquer la fin et le début d’un sous-réseau. Or dans notre étude de cas, le patron (le boss, quoi 🍌) nous demande de créer des sous-réseaux aux masques à longueurs variables. En fait, si on analyse bien la situation, il nous faut créer des sous-réseaux différents dans des sous-réseaux, c’est ce qu’on appelle **subnetter un subnet** (sous-réseauter un sous-réseau 🍌). Il faudra donc, à partir d’un network ID, obtenir un masque différent pour chaque sous-réseau. 🍌 Si nous en étions encore à l’adressage par classes, cela serait impossible car il faut un même masque pour chaque sous-réseau. Ainsi, un réseau tel que 192.168.187.0 n’aurait qu’un seul masque, soit 255.255.255.0.

#### III.6.3.2. TD : implémentation des masques à longueurs variables

Pour comprendre le calcul des masques variables, nous allons implémenter cela dans un réseau local. La solution au scénario ci-dessus se fera comme dans un TD. Les prérequis pour pouvoir suivre et comprendre ce TD sont :

- Maîtrise de la notion de masque de sous-réseau et son utilité.
- Maîtrise des 8 premières puissances de 2.
- Maîtrise de la conversion du binaire au décimal et l’inverse.
- Maîtrise de la notion du subnetting et sa procédure.
- Maîtrise de la notion de passerelle et son rôle.
- Maîtrise de l’adressage CIDR et sa notation.

Pour réussir ce challenge posé par votre patron, il vous faudra suivre des étapes de planification d’adresses. C’est parti ! 🍌

##### III.6.3.2.1. Étape 1 : se focaliser sur le sous-réseau qui a le plus grand nombre d’hôtes

Nous allons commencer par localiser le sous-réseau qui a le plus grand nombre d’hôtes, le sous-réseau A en l’occurrence (50 hôtes).



Combien de bits devons-nous utiliser pour avoir au moins 50 hôtes ?

### III. Veuillez vous identifier pour communiquer

On y va avec les puissances de deux et notre formule  $2^{n-2}$ .

- $2^{1-2} = 0$  ;
- $2^{2-2} = 2$  ;
- .....  
—  $2^{6-2} = 62$ . Stop ! Nous allons donc devoir garder 6 bits de la partie « host » de notre masque.

En binaire, nous obtenons donc : 11000000 . 10101000 . 01100100. nnhhhhhh avec n les bits disponibles pour le réseau, et h les bits **qu'on ne doit pas masquer** pour obtenir au moins 50 hôtes.

#### III.6.3.2.2. Étape 2: choisir un network ID pour l'étape 1

Une fois que nous avons résolu le plus grand sous-réseau, il nous faut choisir quel subnet ID donner à ce sous-réseau. Nous avons retenu, dans l'étape 1, que nous n'avions que 2 bits pour le sous-réseau, ce qui donne (en se focalisant sur le 4e octet) les combinaisons suivantes que vous êtes censé trouver ~~les doigts dans le nez~~ :

- 00hhhhhh ;
- 01hhhhhh ;
- 10hhhhhh ;
- 11hhhhhh.

En remplaçant le h (pour hôte) par 0 (puisque nous ne les masquons pas), on obtient le network ID pour chaque sous-réseau, soit les suivants :

- 00000000 = .0
- 01000000 = .64
- 10000000 = .128
- 11000000 = .192

N'ayant que 2 bits masqués pour le sous-réseau, on peut donc utiliser un masque de /26 ou 255.255.255.192 pour chacun de ses sous-réseaux obtenus, juste comme on ferait dans un réseau utilisant l'adressage par classe. 🍏

Voilà. Nous avons 4 network ID, vous pouvez choisir **n'importe lequel** pour le sous-réseau A. Dans ce TD, nous choisissons au hasard .64, soit la notation **192.168.100.64/26 (network ID/masque)**. Les autres sous-réseaux devront se contenter des trois sous-réseaux restants.



D'où vient le /26 ?

C'est une blague ? 🍏 Vous êtes censés savoir comment on a obtenu le /26. Nous avons gardé 6 bits pour les hôtes, or une adresse IP est constituée de 32 bits, donc  $32-6 = 26$  bits.

### III.6.3.2.3. Étape 3: se focaliser sur le second plus grand sous-réseau

Le second plus grand sous-réseau contient dans notre exemple 27 hôtes pour les développeurs d'applications. Il s'agit du sous-réseau B. Nous allons refaire les calculs de l'étape 1. Nous n'allons donc pas détailler cela. Nous aurons besoin d'au moins 5 bits pour les hôtes ( $2^5 - 2 = 30$ ,  $30 > 27$ , donc O.K.). En binaire, nous avons :

```
11000000 . 10101000 . 11001000 . nnnhhhhh
```

N'est-ce pas ? Alors, nous pouvons faire toutes les combinaisons possibles avec les trois bits pour n, n'est-ce pas ? **Faux !**

?

Faux ? 🍊 Nous n'avons besoin que de 5 bits. C'est logique puisque nous en avons trois consacrés au sous-réseau, non ?

Oui, mais non ! 🍊 Si vous le faites ainsi, vous êtes en train de subnetter le network ID originel soit le 192.168.100.0, alors que c'est ce que nous avons déjà fait dans l'étape 1, en masquant 2 bits pour le réseau. Mais ici, nous voulons **subnetter un subnet**. C'est totalement différent, nous allons donc prendre un sous-réseau déjà obtenu dans l'étape 1 et le re-subnetter à nouveau. 🍊 Avouez que cela devient complexe. 🍊

Dans l'étape 1, nous avons obtenu 4 combinaisons, soit 4 network ID. Nous avons sélectionné le 255.255.255.192 soit le /26 pour le sous-réseau A. Il nous reste donc trois network ID disponibles, soit :

- 00000000 = .0/26
- 01000000 = .64/26 // sous-réseau A
- 10000000 = .128/26
- 11000000 = .192/26

Choisissons le .128 dont le network ID sera **192.168.100.128/26**, ce qui donne en binaire (focus sur le 4e octet) : 10000000. Or nous n'avons besoin que de 5 bits disponibles pour les hôtes, alors qu'ici nous en avons 6. Nous allons donc supprimer un bit pour les hôtes et l'allouer au sous-réseau. 🍊

10n00000.

Voilà ! Nous avons donc 3 bits pour le sous-réseau et 5 pour les hôtes. Maintenant, nous pouvons donc créer deux sous-réseaux à partir du sous-réseau original. 🍊 C'est cela l'intérêt du VLSM, on subdivise encore un sous-réseau. Nous avons donc :

- 10000000 = .128
- 10100000 = .160

Bravo, vous venez de subnetter un subnet comme un pro, vous devez en être fier. 🍊

En résumé, voici ce que nous avons fait :

1. Nous disposions à l'origine d'un network ID de 192.168.100.0/24.
2. Nous l'avons subdivisé pour obtenir un sous-réseau ayant un masque de 192.168.100.64/26 et pouvant contenir au moins 50 hôtes.

### III. Veuillez vous identifier pour communiquer

3. Nous avons pris le sous-réseau obtenu dans l'étape 2 et l'avons *re-subnetté* en deux sous-réseaux (.128 et .160) qui auront un masque de... /27. 🍊

?

Comment avez-vous obtenu le /27 ?

Encore la même question ? 🍊 Dans les deux sous-réseaux que nous avons obtenus, nous avons 3 bits pour les sous-réseaux et 5 pour les hôtes. Nous avons donc laissé 5 bits non masqués pour obtenir au moins 27 hôtes ( $2^5 - 2 = 30$ ). Une adresse IP valant 32 bits,  $32 - 5 = 27$ . 🍊 D'où les réseaux .128 et .160 qui ont tous deux un même préfixe : /27. 🍊

Ici également, nous pouvons choisir n'importe quel network ID : choisissons le premier, soit .128/27. Le sous-réseau restant (.160/27) pourrait être utilisé dans le futur, s'il y a agrandissement du sous-réseau.

Voici la liste des network ID que nous avons obtenus depuis l'étape 1 :

- 00000000 = .0/26 | libre pour être re-subnetté
- 01000000 = .64/26 | sous-réseau A
- 10000000 = .128/26 | Nous ne pouvons plus l'utiliser, il a été re-subnetté
- 10000000 = .128/27 | Nous prenons celui-ci pour le sous-réseau B
- 10100000 = .160/27 | nous gardons celui-ci pour le futur

?

Mais... mais... comment obtient-on 128/26 ET 128/27 ?

C'est faire preuve de concentration que de poser cette question ! La réponse est simple : nous avons 128/26 au départ, mais comme nous l'avons re-subnetté, nous avons changé de masque, en gardant le network ID originel : nous avons donc le 2e 128, mais avec un masque de /27. 🍊

#### III.6.3.2.4. Étape 4: se focaliser sur le troisième plus vaste sous-réseau

Le troisième plus vaste sous-réseau est le réseau C et D de 12 hôtes chacun. Cette étape est exactement la répétition des étapes 1 et 3, avec les mêmes calculs. Nous avons besoin d'au moins 12 hôtes.  $2^4 - 2 = 14$ , ce qui nous suffit. Nous allons donc garder 4 bits pour les hôtes et disposerons donc de 4 bits pour la partie réseau du masque. Au lieu de reprendre le network ID de base et masquer 4 bits comme nous l'aurions fait dans un cas de subnetting classique, nous allons devoir re-subnetter un subnet déjà subnetté comme nous l'avons fait dans l'étape 3. Vous avez le choix : vous pouvez décider de re-subnetter le **192.168.100.0/26**, **192.168.100.128/27** ou **192.168.100.160/27**. Pour ce TD, choisissez **192.168.100.160/27**. Nous avons donc ceci en binaire :

10100000

Nous avons 5 bits libres pour les hôtes, ce qui était suffisant pour le réseau B qui nécessitait 27 hôtes. Mais pour le réseau C ou D, nous aurons une perte de  $27 - 12 = 15$  hôtes, et c'est ce que nous voulons éviter. Nous allons donc retirer un bit de la partie hôte et l'allouer à la partie

### III. Veuillez vous identifier pour communiquer

réseau de notre masque de manière à avoir 4 bits pour les hôtes et 4 pour la partie réseau car  $2^{4-2} = 14$ , ce qui nous convient. 🍊 Nous avons donc :

`101**n**0000`

- **10** représente ici les deux bits consacrés au réseau au départ dans l'étape 1. C'est notre motif de base : nous ne devons pas le changer, le reste de nos subnets en binaire **doit commencer par 00**.
- **1** représente le bit sur lequel nous nous sommes concentrés dans l'étape 3.
- **n** est le bit supplémentaire que nous allons ajouter à la partie réseau du masque de manière à ne rester qu'avec 4 bits pour les hôtes. 🍊

Grace à ce bit de plus que nous avons, nous pouvons donc avoir deux combinaisons (soit le laisser à 0 ou le masquer à 1), ce qui nous donne la possibilité d'obtenir deux autres sous-réseaux à partir du sous-réseau de base, ce qui nous donne (avec le focus sur le 4e octet toujours) :

- `10100000` = .160
- `10110000` = .176

Il ne nous reste plus qu'à trouver le nouveau masque pour ces deux nouveaux sous-réseaux, ce qui est simple puisque nous n'avons qu'à compter le nombre de bits masqués. Soit `11111111.11111111.11111111` puisque nous n'avons que 4 bits pour les hôtes, le reste des bits est donc masqué pour la partie réseau. 🍊 Nous aurons donc un masque de **255.255.255.240** ou **/28** pour la notation CIDR.

Résumons donc tous les subnets obtenus depuis l'étape 1 afin de choisir un network ID pour les réseaux C et D.

- `00000000` = .0/26 | subnet libre pour être re-subnetté
- `01000000` = .64/26 | déjà utilisé par le sous-réseau A
- `10000000` = .128/26 | Nous ne pouvons plus utiliser celui-ci, car nous l'avons re-subnetté
- `11000000` = .192/26 | subnet pour un futur agrandissement
- `10000000` = .128/27 | déjà utilisé pour le sous-réseau B
- `10100000` = .160/27 | nous ne pouvons plus l'utiliser, car nous l'avons re-subnetté
- `10100000` = .160/28 | Prenons celui-ci pour le sous-réseau C
- `10110000` = .176/28 | Prenons celui-ci pour le sous-réseau D
- Remarque 1 : nous nous retrouvons avec deux **160** : un avec **/27** et l'autre avec **/28**, pour les mêmes raisons que nous avons deux **128**. Nous vous avons expliqué cela. 🍊
- Remarque 2 : nous ne pouvons plus utiliser un sous-réseau déjà subnetté.

#### III.6.3.2.5. Cinquième et dernière étape : déterminer les network ID pour les interfaces de liaison

Regardez sur votre schéma. Comment allons-nous nouer ces sous-réseaux entre eux ? Par un routeur bien entendu, et c'est exactement ce que fait le routeur **routeur\_link** dans le schéma. Mais comme chaque routeur, il possède deux interfaces de liaison. Il faut donc deux adresses IP pour relier F et E, deux pour relier G et E, et deux pour relier H et E. 🍊 Nous pouvons le faire les yeux fermés en binaire, nous n'avons besoin de 2 bits car  $2^{2-2} = 2$ . Ça tombe pile-poil. 🍊 Nous devons donc choisir un sous-réseau libre pour être re-subnetté. Vous pouvez choisir celui que vous voulez parmi les sous-réseaux libres que nous avons obtenu jusqu'ici. Pour suivre ce TD, choisissez le sous-réseau `00000000/26`. Ici également, nous nous rendons compte

### III. Veuillez vous identifier pour communiquer

que ce sous-réseau a été subnetté de manière à obtenir au moins 50 hôtes (sous-réseau A), voilà pourquoi 6 bits sont disponibles pour les hôtes. Or nous n'avons besoin que de deux bits. Qu'allons-nous faire ? Exactement ce que nous avons fait dans les étapes précédentes, nous allons allouer 4 bits de plus à la partie réseau du masque pour ne rester qu'avec deux bits pour les hôtes. 🍊 Nous obtenons donc ceci :

00\*\*\*nnnn\*\*hh

Maintenant nous avons 4 bits alloués au réseau, ce qui donne la possibilité d'obtenir 16 sous-réseaux ( $2^4$ ) à partir de ce sous-réseau. Vous devez être capable de déterminer chacun de ces 16 sous-réseaux, mais comme nous sommes gentils aujourd'hui, nous vous écrivons **les huit premiers, l'avant-dernier et le dernier**. 🍊 Nous avons donc :

```
— 00000000 = .0/30
— 00000100 = .4/30
— 00001000 = .8/30
— 00001100 = .12/30
— 00010000 = .16/30
— 00010100 = .20/30
— 00011000 = .24/30
— 00011100 = .28/30
— 00100000 = .32/30
— .....
— .....
— 00111000 = .56/30
— 00111100 = .60/30
```



N.B. : chacun de ces sous-réseaux nous donnera deux hôtes disponibles, vous pouvez donc choisir quatre sous-réseaux parmi les 16 pour les interfaces de liaison, c'est-à-dire pour les adresses IP de chaque interface du routeur. 🍊 Nous allons choisir les quatre premiers, soit de .0/30 -.12/30.

Tenez, si on vous demandait d'agréger les 12 sous-réseaux restants (soit à partir de .16/30) en utilisant la technique du supernetting pour obtenir une seule route, laquelle obtiendrez-vous ?

Un indice ?

👁 Contenu masqué n°5

Ne regardez la réponse qu'après avoir vraiment essayé de trouver : cela ne vous aide pas si vous ne faites pas un effort de pratiquer. 🍊

👁 Contenu masqué n°6

Résumons chacun des sous-réseaux trouvés depuis l'étape 1 :

### III. Veuillez vous identifier pour communiquer

- 00000000 = .0/26 | nous ne pouvons plus utiliser celui-ci, nous l'avons re-subnetté
- 00000000 = .0/30 | utilisons celui-ci pour le sous-réseau E
- 00000100 = .4/30 | sous-réseau F
- 00001000 = .8/30 | sous-réseau G
- 00001100 = .12/30 | sous-réseau H
- 01000000 = .64/26 | sous-réseau A
- 10000000 = .128/26 | Nous ne pouvons plus utiliser celui-ci, car nous l'avons re-subnetté
- 11000000 = .192/26 | subnet pour un futur agrandissement
- 10000000 = .128/27 | sous-réseau B
- 10100000 = .160/27 | nous ne pouvons plus l'utiliser, car nous l'avons re-subnetté
- 10100000 = .160/28 | sous-réseau C
- 10110000 = .176/28 | sous-réseau D

#### III.6.3.3. Conclusion et exercices

Ce TD vous a certainement aidé à mieux assimiler le principe du VLSM et du Supernetting. Nous avons au passage revu un certain nombre de notions. En effet, nous avons pratiqué les calculs des puissances de 2, la conversion des masques du décimal au binaire et vice versa, le subnetting d'un réseau, l'interconnexion de deux sous-réseaux différents par le biais d'un routeur etc. Plusieurs autres cas de pratique viendront avec le temps pour vous faire pratiquer tout ce que nous avons appris depuis la première partie du cours : le design de l'infrastructure d'un réseau, l'analyse des contraintes et la mise en place de la meilleure infrastructure répondant au mieux aux besoins, la segmentation des réseaux selon des exigences, la planification d'un plan d'adressage, le diagnostic et bien d'autres joyeusetés qui vous donneront indubitablement un savoir-faire dans le domaine, savoir-faire assez solide pour être admis en stage d'entreprise dans le domaine du réseau. 🍊

Pour terminer, nous vous donnons quelques exercices, pas difficiles du tout.

Dans ce TD, nous n'avons fait que trouver les network ID de chaque subnet qui répondaient aux consignes de notre patron, ce qui, mine de rien, représente plus de 50 % du travail. 🍊 À présent, votre travail consistera à la détermination des plages d'adresses pour chaque sous-réseau. Voici, en résumé, ce que vous devez faire :

- Pour chaque sous-réseau (A, B, C, D, E, F et G), déterminez les plages d'adresses. Dans ces plages, déterminez celles que vous allez assigner aux hôtes. Par exemple pour le réseau C, nous aurons 14 adresses utilisables, mais seulement 12 assignables car notre réseau ne comprend que 12 hôtes ; les deux autres seront une perte, certes, mais réduite au maximum.
- Choisissez la première adresse IP de chaque plage pour le routeur.
- Éditez le schéma que vous avez utilisé pour ce TD et inscrivez les adresses IP de chacune des interfaces de chaque routeur ainsi que leurs masques respectifs (notation avec un / exigée), pour obtenir quelque chose de similaire au schéma **Figure 1.0** (les adresses IP sont fictives et nous avons omis les masques). Si vous ne pouvez pas l'éditer, contentez-vous de résumer cela dans un tableau. 🍊
- Ajoutez trois hôtes dans les réseaux A, B, C et D et inscrivez, à côté de chacun d'eux, leurs adresses IP et masques respectifs. Votre schéma doit ressembler au schéma **Figure**

### III. Veuillez vous identifier pour communiquer

- 1.1. (Ici également, les adresses IP sont fictives et nous nous limitons à un hôte par réseau pour gagner le temps. 🍏 )
- Finalement, faites communiquer les hôtes de chaque réseau entre eux. C'est-à-dire, un hôte A1 du réseau A doit communiquer avec un hôte du réseau B, un autre du réseau C et un autre du réseau D. Vous n'avez pas besoin d'un logiciel de simulation pour le faire. Mettez juste des flèches pour illustrer le cheminement du message envoyé, ensuite déterminez les routes (ou les adresses IP des interfaces) par lesquelles ce message passera pour arriver à son destinataire. Votre schéma doit ressembler au schéma Figure 1.2 (ici également, les adresses sont fictives et nous n'avons pas mis les masques de chaque adresse). Vous pouvez, bien entendu, résumer cela dans un tableau à défaut du schéma.

#### III.6.3.4. Ci-dessous les images d'illustration

Figure 1.0 - Les réseaux de zéro - zestedesavoir.com

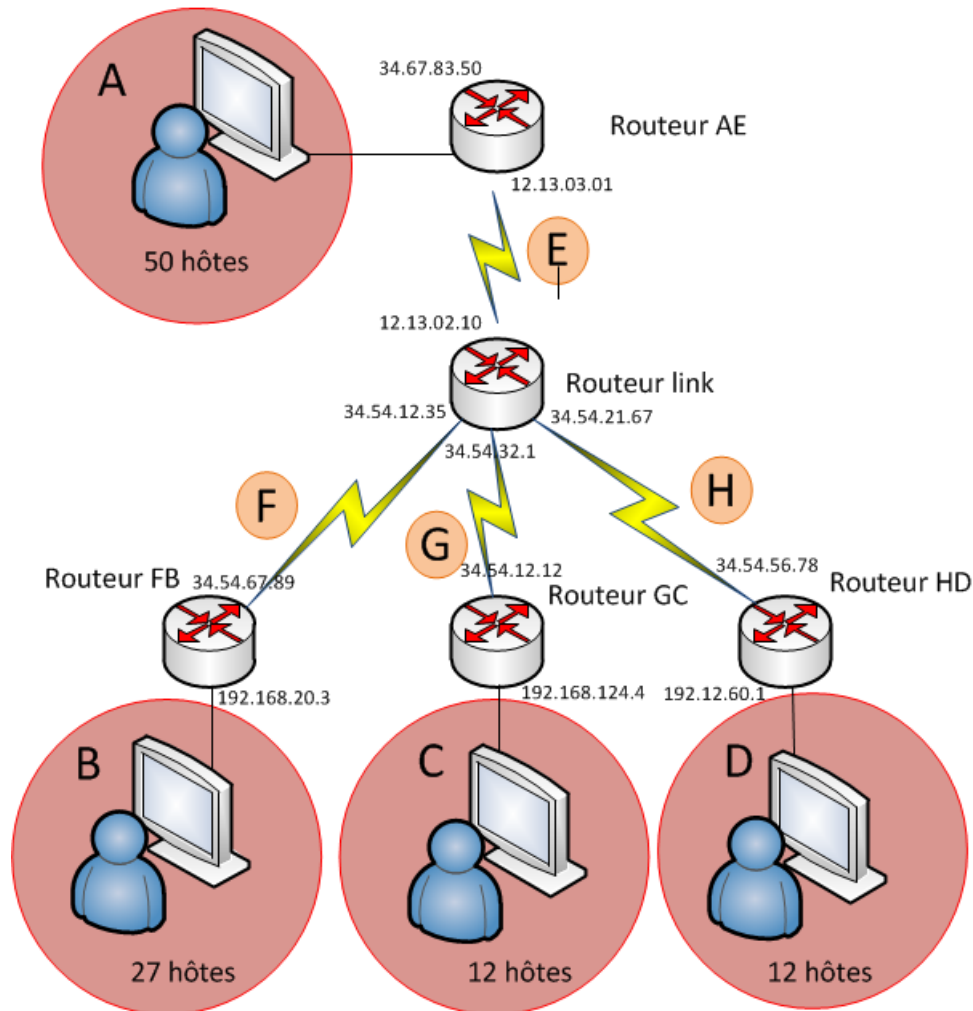


FIGURE III.6.2. – Figure 1.0

Figure 1.1 - Les réseaux de zéro - zestedesavoir.com

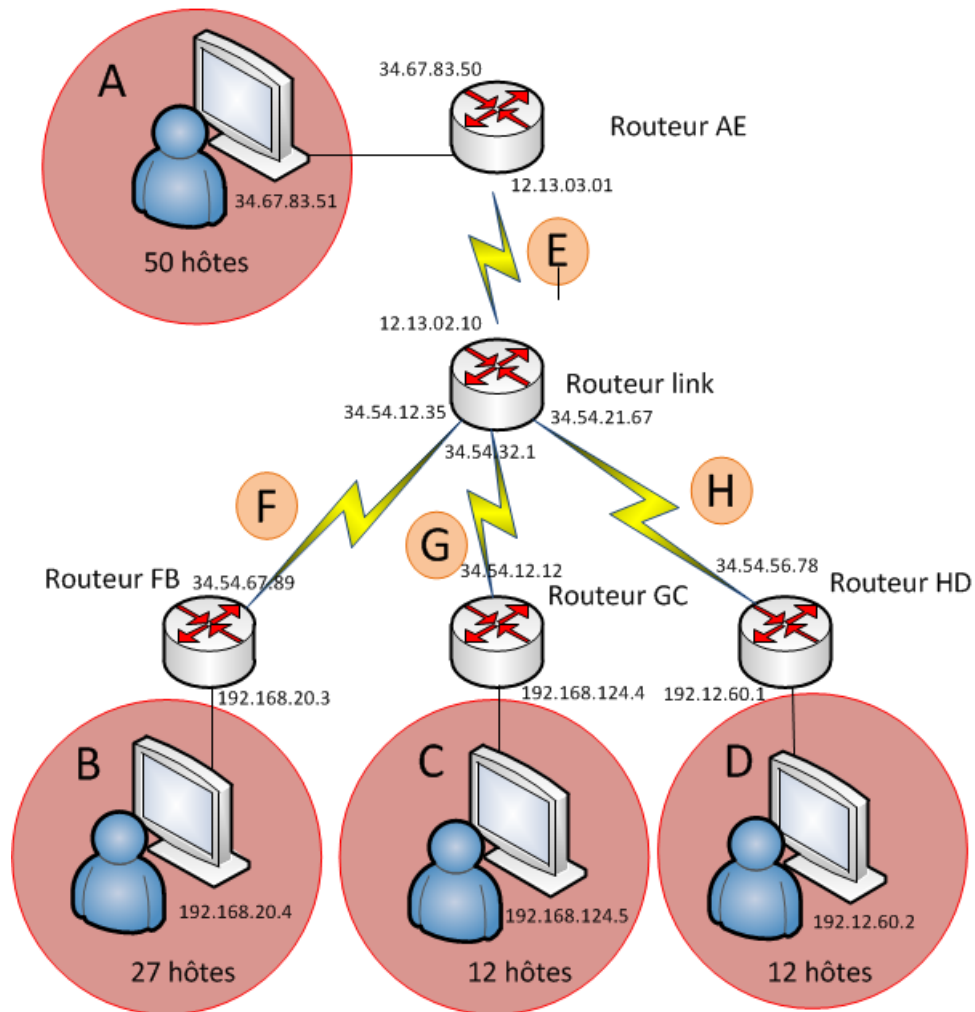


FIGURE III.6.3. – Figure 1.1

Figure 1.2 - Les réseaux de zéro - zestedesavoir.com

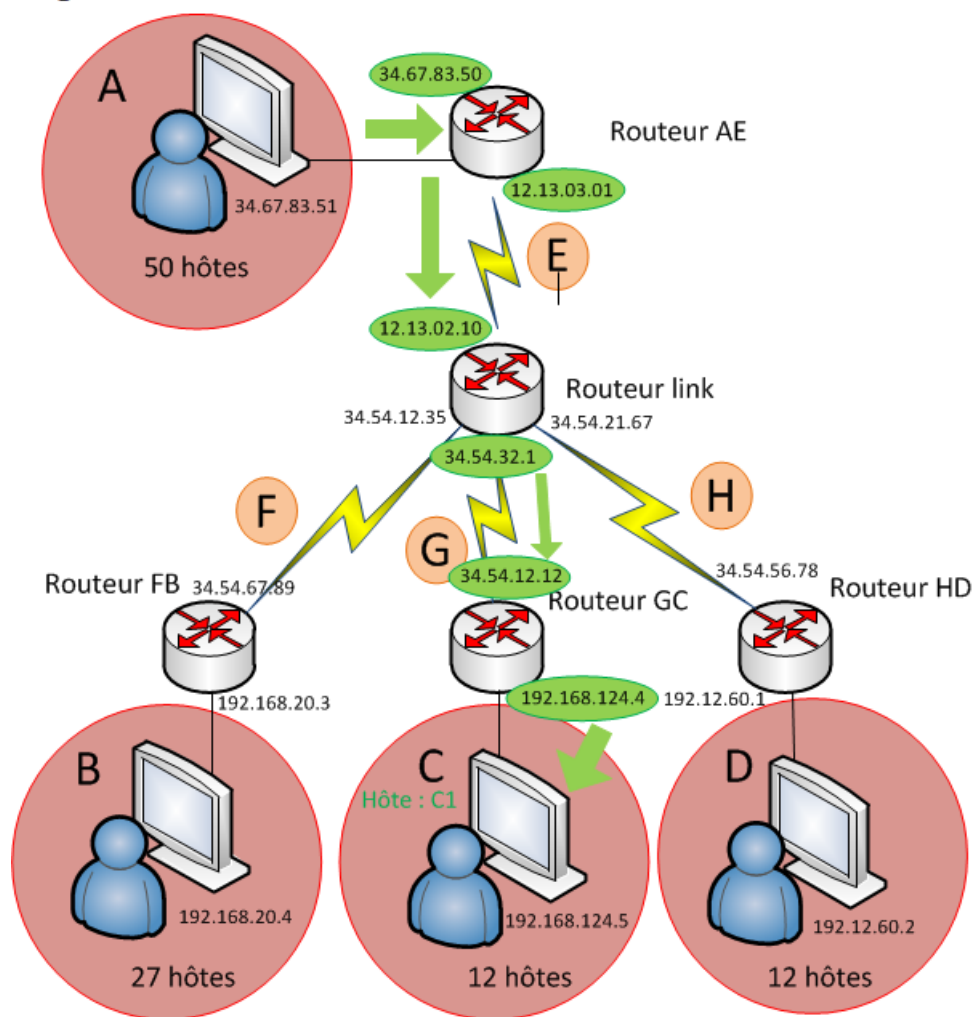


FIGURE III.6.4. – Figure 1.2

Voilà ! Lorsque vous aurez fini, vous pourrez poster vos solutions sur le forum.

i

Il est vraiment important de réussir ces exercices tout seul.

## Conclusion

Nous espérons que vous avez compris tout l'intérêt de l'adressage CIDR. Nous escomptons que cela vous paraît évident maintenant que vous avez constaté par vous-même la différence entre l'adressage par classe et l'adressage CIDR.

Maintenant que vous avez bien tout appris concernant les adresses IPv4, passons à IPv6 ! 🍊

## Contenu masqué

### Contenu masqué n°4

Voici trois possibilités de résoudre l'exercice 2.

Cas 1. On veut juste déterminer les sous-réseaux et plages associées pour le nombre d'hôtes et de serveurs donnés. Dans ce cas, on commence par trier les sous-réseaux par nombre d'adresses souhaitées : netcomp (76 clients + 1 serveur), netprog (47 clients + 1 serveur), netbackup (36 clients + 1 serveur), netsecr (33 + 1).

On commence donc par netcomp. Il va nous falloir un sous-réseau avec 77 adresses minimum. On cherche la puissance de deux supérieure.  $2^6 = 64$ ,  $2^7 = 128$ , on va prendre 128 bits. Pour netcomp, on réserve donc 7 bits pour le host id, ce qui nous fait un masque en /25, soit 11111111.11111111.11111111.10000000. Le network id sera le premier disponible, à savoir 120.12.0.0. L'adresse du premier hôte sera 120.12.0.1 et le dernier 120.12.0.126 (on gardera 120.12.0.127 pour le broadcast).

Pour netprog, il nous faut 48 adresses. Si on repart de la fin du sous-réseau précédent, on peut refaire un sous-réseau en exploitant les adresses en 120.12.0.xxx inutilisées (de 128 à 255). On ne pourra toutefois pas aller plus loin : il est mathématiquement impossible de trouver une adresse de réseau et un masque qui permettraient de couvrir les adresses de 120.12.0.128 à 120.10.1.255 (par exemple) sans englober aussi le premier sous-réseau. Procédons comme pour netcomp en trouvant une puissance de 2 supérieure à 48.  $2^5 = 32$ ,  $2^6 = 64$ . On va donc prendre 6 bits, ce qui nous fait un masque en /26 et une plage allant de 120.12.0.128 à 120.12.0.191. La première adresse d'hôte est 120.12.0.129, la dernière 120.12.0.190, le broadcast 120.12.0.191.

On peut faire exactement pareil pour netbackup, on trouvera 120.12.0.193 pour le premier hôte, 120.12.0.254 pour le dernier, 120.12.0.255 pour le broadcast.

Enfin, pour le dernier, on fait la même chose : on part de l'adresse suivante disponible (120.12.1.0), on prend 6 bits et on a une plage 120.12.1.0 - 120.12.1.63.

On a optimisé au maximum les adresses à notre disposition, ce qui nous en laisse plein pour d'autres usages. Cette méthode a un inconvénient : elle limite l'extension des sous-réseaux. Si on veut rajouter 20 hôtes à netprog, ça en fera 68 et tout notre plan sera fichu ! Mais on peut aussi procéder d'une autre manière, en considérant qu'on n'aura jamais besoin de créer d'autres sous-réseaux et qu'on veut juste de la place pour ces quatre sous-réseaux là.

Cas 2. Dans ce cas, on peut couper le réseau en quatre parts égales. Nos adresses vont de 120.12.0.0 à 120.12.63.255. On peut ainsi définir arbitrairement les plages suivantes : 120.12.0.0 - 120.12.15.255, 120.12.16.0 - 120.12.31.255, 120.12.32.0 - 120.12.47.255, 120.12.48.0 - 120.12.63.255. C'est beaucoup plus facile mais moins souple : en cas de création de nouveaux sous-réseaux, il faudra tout revoir.

Cas 3. Heureusement, une méthode de super fainéant existe ! 🍊 Elle combine les avantages des deux autres méthodes, à savoir la flexibilité pour ajouter des sous-réseaux et pour ajouter des hôtes. Il suffit de se dire qu'on va mettre 256 adresses pour tout le monde ! Pourquoi 256 ? Parce que c'est suffisant pour tous et ça laisse de la marge de manœuvre. Et surtout, c'est hyper facile à calculer : du /24 pour tous, et on a netcomp : 120.12.0.0–255, netprog : 120.12.1.0–255,

### *III. Veuillez vous identifier pour communiquer*

netbackup : 120.12.2.0–255, netsecr : 120.12.3.0–255. Et voilà, une demi-heure de gagnée, ça fait une pause café bien méritée ! 🍊

[Retourner au texte.](#)

#### Contenu masqué n°5

Suivez les étapes expliquées dans la sous-partie précédente.

[Retourner au texte.](#)

#### Contenu masqué n°6

00010000 = .16/26 ou 255.255.255.192

[Retourner au texte.](#)

## III.7. Et IPv6 dans tout ça ?

### Introduction

Vous en avez bavé avec les adresses privées, le *subnetting*, les VLSM, ... ? Rassurez-vous, tout cela n'existera plus d'ici quelque temps ! 🍊

?

QUOI ? On s'est pris la tête pour rien ?

En fait, non. Tout au long de cette partie, nous avons travaillé sur IPv4. Cela fait des années qu'on entend dire que ça va disparaître au profit d'IPv6. Cela fait des années que tout le monde utilise encore IPv4. Pas de panique, donc. 🍊

Toutefois, il est inévitable qu'un jour, on délaisse IPv4 au profit d'IPv6. En pratique, cela ne devrait pas changer grand-chose pour les particuliers. Ce qui choquera le plus, ce sera de voir que nos sempiternels 192.168.1.1/24 seront remplacés par de monstrueux FE80::68AF:ADFF:3E80:B69D%11 (au hasard). C'est pourquoi dans ce chapitre, nous allons découvrir les adresses IPv6 qui, malgré leur apparence, ne sont pas plus compliquées que les adresses IPv4.

!

Pour pouvoir suivre ce chapitre, vous devez connaître le binaire et l'hexadécimal. Si vous ne maîtrisez pas ces sujets, [consultez cette annexe](#) 🔗 avant de vous investir dans les adresses IPv6.

### III.7.1. Un format particulier

Commençons par une bonne nouvelle : vous avez probablement déjà une adresse IPv6.

?

Ah bon ? 🍊

Eh oui ! Tapez la commande `ipconfig` (Windows) ou `ip a` (Linux / Mac OS) dans une console. Sur Android, vous pouvez vous rendre dans les paramètres et rechercher "adresse IP". Dans les informations de votre carte réseau, vous devriez trouver quelque chose qui ressemble à ça :

```
1 Adresse IPv6 de liaison locale. . . . .:
    fe80::8c79:7ab:73ea:9943%11
```

### III. Veuillez vous identifier pour communiquer

Ou encore :

```
1 inet6addr: fe80::8c79:7ab:73ea:9943/64 Scope: link
```

Il se peut même que vous en ayez plusieurs, nous allons y revenir.

?

Bouh, je n'ai pas d'adresse IPv6, que vais-je faire ? 🍌

Dans ce cas, il est probable que vous ne soyez connecté à aucun réseau. Dès lors que vous êtes connecté en Wi-Fi, 5G, par câble ou autre, peu importe le réseau, vous devriez voir apparaître une adresse IPv6. Une autre raison possible : votre système est suffisamment ancien pour ne pas prendre en charge IPv6. Si tel est le cas, c'est qu'il est grand temps d'en changer : tous les systèmes le supportent depuis la fin des années 2000 ! 🍌

Ça y est ? Vous avez une adresse IPv6 ? Maintenant, nous allons apprendre à la lire !

#### III.7.1.1. Lire une adresse IPv6

Cela vous a sans doute frappé, nous avons affaire à quelque chose de difficilement lisible. En IPv4, il y a 4 nombres décimaux inférieurs à 255 séparés par des points. Une adresse prend donc 4 octets (32 *bits*). En IPv6, il y a 8 nombres hexadécimaux inférieurs à FFFF séparés par le symbole « : », cela prend 16 octets (128 *bits*).

?

8 nombres ? Dans l'exemple donné, je n'en compte que 5 !

Si déjà, vous n'en comptez que 5 et pas 6, c'est que vous avez deviné que les %11 et /64 étaient des masques de sous-réseau. Bravo !

Effectivement, on ne lit que 5 nombres. Si vous êtes attentifs, vous remarquerez qu'après fe80, il y a deux fois le symbole « : ». Retenez cette explication, c'est fondamental :

**Dans une adresse IPv6, « :: » signifie « remplir avec autant de zéros qu'il faut pour avoir 128 *bits* ».**

Reprenons l'adresse précédente et comptons le nombre de *bits* de chaque bloc.

<b>FE80</b>	<b>::</b>	<b>8C79</b>	<b>7AB (</b> <b>=</b> <b>07AB)</b>	<b>73EA</b>	<b>9943</b>
16 <i>bits</i>	?	16 <i>bits</i>	16 <i>bits</i>	16 <i>bits</i>	16 <i>bits</i>

L'ordinateur calcule tout seul le nombre de *bits* que représente le « :: ». Ici, il représente 128 - (16 + 16 + 16 + 16 + 16) = 48 *bits*. L'adresse fe80::8c79:7ab:73ea:9943 peut donc aussi s'écrire fe80:0:0:0:8c79:7ab:73ea:9943.

### III. Veuillez vous identifier pour communiquer

Vous remarquerez qu'en hexadécimal comme en décimal, on n'écrit pas les zéros inutiles. On pourrait écrire l'adresse précédente sous la forme `fe80:0000:0000:0000:8c79:07ab:73ea:9943`, mais quel intérêt ? Ce serait comme écrire `010.000.050.188` au lieu de `10.0.50.188`. Chaque bloc occupe toujours 16 *bits*, que l'on écrive les zéros inutiles ou pas. Regardez cet autre exemple : l'adresse `FE80::1234:1`.

<b>FE80</b>	<b>::</b>	<b>1234</b>	<b>1</b>
16 <i>bits</i>	80 <i>bits</i>	16 <i>bits</i>	16 <i>bits</i>

Le 1 à la fin occupe 16 *bits*. Ce nombre seul pourrait s'écrire avec un seul chiffre binaire, mais dans une adresse IPv6, il constitue un bloc de 16 *bits* à lui tout seul, au même titre que FE80 et 1234.



On ne peut utiliser le double deux-points qu'une fois par adresse IPv6. Pour abréger l'adresse `fe80:0:0:0:a4:0:0:1`, on peut écrire `fe80::a4:0:0:1` ou bien `fe80:0:0:0:a4::1`, mais **en aucun cas**, on ne peut écrire `fe80::a4::1`. C'est normal : à partir d'une telle adresse, comment peut-on deviner combien de zéros il faut mettre à la place des :: ? Faudrait-il mettre 4 blocs de zéros en premier et un seul en second lieu ? Ou bien 3 d'abord puis 2 ensuite ? Une telle écriture est donc impossible.

Voilà, vous savez lire une adresse IPv6 ! C'était pas si difficile que ça, si ? 🍊 Une dernière chose : pour noter un masque de sous-réseau, on peut utiliser le slash (/) ou le pourcentage (%). Selon les systèmes d'exploitation, vous rencontrerez l'un ou l'autre.

Vous avez peut-être noté que nous avons dit en début de chapitre que vous avez peut-être plusieurs adresses IPv6. Effectivement, une carte réseau peut avoir plusieurs adresses IPv6 simultanément. Chacune d'entre elles correspond à un usage précis. C'est l'objet du reste de ce chapitre.

#### III.7.2. L'adresse privée de sortie

Quelque chose a peut-être attiré votre attention tout à l'heure lorsque nous avons cherché si nous avions déjà une adresse IPv6 : l'expression « adresse de liaison locale ». Ce genre d'adresse commence forcément par FE80::/10. Qu'est-ce que cela signifie ?

$$\text{FE80}_{(16)} = \mathbf{1111\ 1110\ 10\ 00\ 0000}_{(2)}$$

Cela signifie que les 10 premiers *bits* d'une adresse de lien local valent obligatoirement 1111111010. Voici donc quelques exemples d'adresses de lien local :

- FE80::1 (**1111 1110 10 00 0000** [111 zéros] 1)
- FE85::2:1:dada (**1111 1110 10 00 0101** [78 zéros] 10 0000 0000 0000 0001 1101 1010 1101 1010)
- FE93::3 (**1111 1110 10 01 0011** [110 zéros] 11)

Par contre, ces exemples ci-dessous **ne peuvent pas** être des adresses de lien local :

### III. Veuillez vous identifier pour communiquer

- FEC0::1 (1111 1110 11 00 0000 [111 zéros] 1)
- FF00::1 (1111 1111 00 00 0000 [111 zéros] 1)
- ABCD::EFAC (1010 1011 11 00 1101 [96 zéros] 1110 1111 1010 1100)

La particularité de cette adresse est qu'elle ne peut être utilisée que pour contacter un hôte directement connecté. Les exemples suivants ont été réalisés avec le logiciel *Packet Tracer*.



FIGURE III.7.1. – 2 ordinateurs sont connectés directement

Depuis l'ordinateur 1 (FE80::1), on « pingue » le 2 (FE80::2). Résultat :

```
1 PC>ping fe80::2
2
3 Pinging fe80::2 with 32 bytes of data:
4
5 Reply from FE80::2: bytes=32 time=11ms TTL=128
```

Aucun souci. Maintenant, on relie les 2 ordinateurs au routeur, comme ceci :

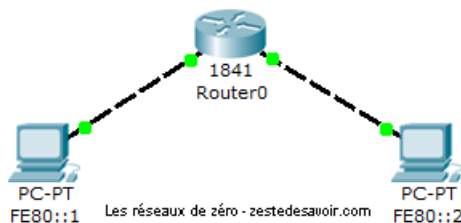


FIGURE III.7.2. – 2 ordinateurs reliés par un routeur

Sachez que l'interface du routeur reliée au PC 1 a pour adresse FE80::3 et l'autre FE80::4.

On relance la même commande :

```
1 PC>ping fe80::2
2
3 Pinging fe80::2 with 32 bytes of data:
4
5 Request timed out.
```

Il n'y a pas d'histoire de masque de sous-réseau, la preuve :

### III. Veuillez vous identifier pour communiquer

```
1 PC>ping fe80::4
2
3 Pinging fe80::4 with 32 bytes of data:
4
5 Request timed out.
6
7 [...]
8
9 PC>ping fe80::3
10
11 Pinging fe80::3 with 32 bytes of data:
12
13 Reply from FE80::3: bytes=32 time=5ms TTL=255
```

Avec des adresses de lien local, on ne peut donc vraiment que contacter un hôte auquel on est directement connecté !

Les adresses des exemples ci-dessus ont été attribuées manuellement (pour qu'elles soient courtes et donc plus faciles à taper). 🍊 Mais sur votre machine, vous n'avez pas choisi votre adresse IPv6 : elle a été attribuée automatiquement ! Pour obtenir une adresse IPv6, il y a plusieurs manières. Soit il y a un serveur DHCPv6 qui distribue des adresses à tour de bras sur le réseau et on prend ce qu'il donne. Soit il n'y en a pas et votre ordinateur la génère lui-même.

Pour générer une adresse IPv6, certains systèmes comme Windows 7 tirent des nombres au hasard et paf ! Ça fait ~~des chœpies~~ une adresse. D'autres, comme Windows XP, se basent sur l'adresse MAC de la carte réseau : ils changent d'état le 7ème *bit*, insèrent FFFE en plein milieu et collent le tout après FE80::.

On ne peut donc pas communiquer avec des hôtes éloignés en utilisant les adresses de lien local. Qu'à cela ne tienne ! Pour aller sur Internet, nous utiliserons des adresses globales.

#### III.7.3. Les adresses qui voyagent

Les adresses IPv6 utilisées sur Internet sont très structurées et hiérarchisées. Regardez le tableau ci-dessous :

IANA	RIR	LIR	Client	Sous-réseau	Hôte
3 bits	20 bits	9 bits	16 bits	16 bits	64 bits

À l'époque où cela a été conçu, l'IANA était l'organisme qui gérât les adresses IP sur Internet. Il a décidé que les adresses IPv6 utilisables sur Internet commenceraient par 2000::/3, puis il donne au RIR des valeurs sur 20 *bits* pour différentes zones géographiques. Chaque RIR donne ensuite aux LIR des valeurs pour chacun d'entre eux, qui eux-mêmes vont distribuer des valeurs aux clients, qui vont... etc.



IANA ? RIR ? LIR ? Euh... ? 🍌

L'IANA, c'est le chef. Ou plutôt, c'était le chef jusqu'en 2016, quand l'ICANN a repris ce rôle. Il existe différents RIR pour différentes zones géographiques : l'ARIN pour l'Amérique du Nord, l'AfriNIC pour l'Afrique, l'APNIC pour l'extrême-orient et l'Océanie, etc. Le chef a dit : tel RIR prendra telle valeur pour son champ. Ainsi, toutes les adresses IPv6 japonaises commencent par 2001:0200::/23, car l'APNIC (qui couvre le Japon) a obtenu la valeur 00000000000010000001<sub>(2)</sub>.

Les LIR, ce sont les fournisseurs d'accès à Internet et les très grosses entreprises. Par exemple, OVH détient tout le bloc 2001:41d0::/32 (si on fait le calcul, il a obtenu la valeur 111010000<sub>(2)</sub>). SFR détient le bloc 2001:4c18::/32. Ces informations peuvent être consultées [sur cette page](#) ↗.

Ensuite, chacune de ces entreprises dispose en théorie de 65536 adresses pour ses clients. En réalité, elle peut empiéter sur le champ « sous-réseau » pour étendre ce nombre et laisser les clients faire des sous-réseaux en empruntant des *bits* du champ « hôte » (il y a de quoi faire!).

FE80::/10, 2000::/3... Cela fait quand même un paquet d'adresses inutilisées ! Que deviennent les autres ? Bonne question... De grands blocs sont réservés pour des usages privés. D'autres sont réservés pour des usages spécifiques. C'est ce que nous allons voir maintenant !

## III.7.4. Les adresses particulières

En dehors des adresses de lien local et globales, il existe aussi quelques adresses particulières à connaître.

### III.7.4.1. Les adresses courtes

Celles-là sont appréciées pour leur facilité d'écriture. 🍌 On trouve 2 adresses particulièrement courtes :

- **::1** : c'est l'adresse de la boucle locale (loopback). C'est l'équivalent de 127.0.0.1 en IPv4.
- **::** (ou **::0**) : c'est l'adresse que prend votre carte réseau avant de s'attribuer une adresse. Quand votre machine envoie une demande d'adresse au démarrage, elle le fait sous l'adresse ::. Nous aurons l'occasion d'y revenir quand nous parlerons de la couche réseau du modèle OSI.

### III.7.4.2. Les adresses de site local

Ces adresses sont attribuées au sein d'un site mais ne sont pas accessibles depuis le réseau public (comme les adresses privées en IPv4). Elles commencent par FEC0::/10 et sont structurées de la manière suivante :

10 bits	54 bits	64 bits
---------	---------	---------

### III. Veuillez vous identifier pour communiquer

1111 1110 11	identifiant du sous-réseau	identifiant de l'hôte
--------------	-------------------------------	-----------------------

Ces adresses peuvent être routées (elles peuvent passer des routeurs) dans le site local, mais pas au delà.

#### III.7.4.3. Les adresses multicast

Elles commencent par FF00::/8 et servent à désigner et gérer des groupes d'hôtes. Le multicast n'est pas l'objet de ce chapitre, il y a tellement de choses à dire à ce sujet que nous y reviendrons ultérieurement. Sachez juste, pour le moment, qu'une adresse IPv6 qui commence par FF ne désigne pas un hôte.



Contrairement à IPv4, en IPv6, il n'existe pas d'adresse de *broadcast* !

#### III.7.4.4. Les adresses d'encapsulation 4/6

IPv6 est compatible avec IPv4. On peut s'adresser à une adresse IPv4 même si on a juste une adresse IPv6. Comment faire alors pour désigner une adresse IPv4 en IPv6 ? En la représentant de cette manière :

10 octets	2 octets	4 octets
Que des zéros	FFFF(16)	L'adresse IPv4

Voici quelques exemples :

Adresse IPv4	Adresse IPv6 d'encapsulation
42.13.37.42	::FFFF:42.13.37.42
192.168.2.1	::FFFF:192.168.2.1

La notation peut surprendre au premier abord. Pourtant, elle est courante et est plus facile à comprendre que si on l'écrivait entièrement en hexadécimal.

Vous savez maintenant différencier les différentes adresses IPv6 !

## Conclusion

Voilà pour ce tour d'horizon des adresses IPv6. Nous n'avons parlé que du format des adresses mais pas de ses avantages : nous verrons cela quand nous traiterons la couche 3 du modèle OSI. Mais avant, il y a la couche 4 !

# Conclusion

Maintenant que vous avez les connaissances nécessaires pour comprendre ce qui va suivre, plus rien ne vous empêche de foncer ! Néanmoins, comme vous avez pu le constater, les chapitres deviennent de plus en plus consistants. Aussi, il vaut mieux que vous alliez doucement et que vous preniez le temps de bien tout assimiler, sinon, vous risquez de vous perdre !

## Quatrième partie

Dans les basses couches du modèle OSI

# Introduction

Reprenons notre descente des couches du modèle OSI ! Attention, ça devient un peu plus lourd, ne vous précipitez pas !

## IV.1. Introduction à la couche transport

### Introduction

Les couches transport et réseau sont les plus importantes du modèle OSI. Nous allons donc les examiner en détail. Ce chapitre sert d'introduction à l'univers de la couche de transport.

#### IV.1.1. Présentation

Le rôle de la couche transport est de **rendre possible la communication logique entre applications**. En anglais, on parle de *logical end-to-end communication*.



Pourquoi « logique » ?

Une communication logique, c'est une communication « distante », dans le sens où les hôtes communiquant ne sont pas forcément reliés physiquement, mais ils ne s'en rendent pas compte. Pour eux, ils sont connectés et peuvent communiquer, c'est tout ce qu'ils savent. Le reste, ils s'en fichent !

La couche transport établit une communication logique entre les **processus d'applications**. Cela signifie qu'un programme, une application sur votre ordinateur utilise la couche 4 pour établir une transmission avec une autre application. Seule cette communication lui est connue, tous les autres échanges de votre machine lui sont hors de portée (et accessoirement, elle n'en a rien à faire 🍊).

Prenons deux hôtes A et B qui s'échangent des fichiers avec le protocole BitTorrent, que nous avons vu précédemment. Ils utilisent chacun un client pour partager leurs torrents. Donc il s'agit d'un même type d'application dans la transmission et la réception. Voilà pourquoi on parle de *end-to-end application communication*, ce qui en français peut se traduire par « bout-en-bout ». Par « bout-en-bout », la pensée exprimée est que la communication entre les couches se fait de façon parallèle.

Voici un schéma illustrant ce principe de communication parallèle :

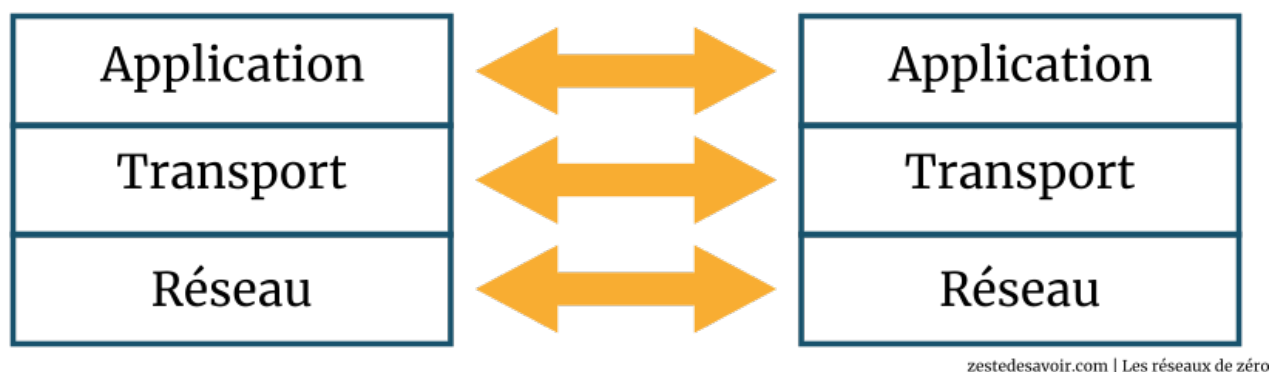


FIGURE IV.1.1. – Communication de bout-en-bout (CC BY)

Ce principe reste valable pour les autres couches du modèle OSI ou TCP-IP. Chaque couche communique parallèlement avec sa couche « homologue ». La couche application de l'hôte émetteur communique directement avec la couche application de l'hôte récepteur, de même que la couche transport de l'émetteur communique directement avec la couche transport de l'hôte récepteur.

Si c'est un peu confus, référez-vous à [la section traitant du principe d'encapsulation](#) . Si vous vous souvenez bien, nous avons vu que la couche réseau, par exemple, ajoutait un en-tête au SDU de la couche transport, le transformant ainsi en un PDU. Une fois au niveau de la couche réseau du récepteur, cet en-tête était lu et supprimé. Ces deux couches (réseau) communiquaient donc parallèlement.

#### IV.1.1.1. L'histoire de Couic le message

Le rôle de la couche transport est bien sûr d'assurer le transport des messages, mais parfois, ces derniers sont trop longs. Pour faire une analogie, une voiture sert au transport routier, mais on ne peut pas y faire rentrer des objets forts volumineux. Celles et ceux qui ont essayé de déménager un lit avec une citadine savent bien de quoi on parle. 🍊 Eh bien en réseau, l'avantage, c'est qu'on peut découper quand ça ne rentre pas !

Quand l'hôte A envoie un message, il utilise une application comme un client de messagerie. Cette application lui donne accès aux services réseaux pour envoyer un mail, donc le protocole SMTP sera utilisé, par exemple. Une fois ce SDU reçu par la couche transport, il sera converti en un PDU, comme nous l'avons vu. La couche transport, qui est aussi responsable de la fragmentation des unités de données, va « couper » ce SDU en plusieurs chaînes (ou morceaux) qu'on appelle « *chunks* » en anglais. À chaque « morceau » sera ajouté un en-tête. Le SDU reçu par la couche transport sera donc « brisé » en 4 PDU distincts, par exemple. 🍊

Voici deux schémas illustrant cela :

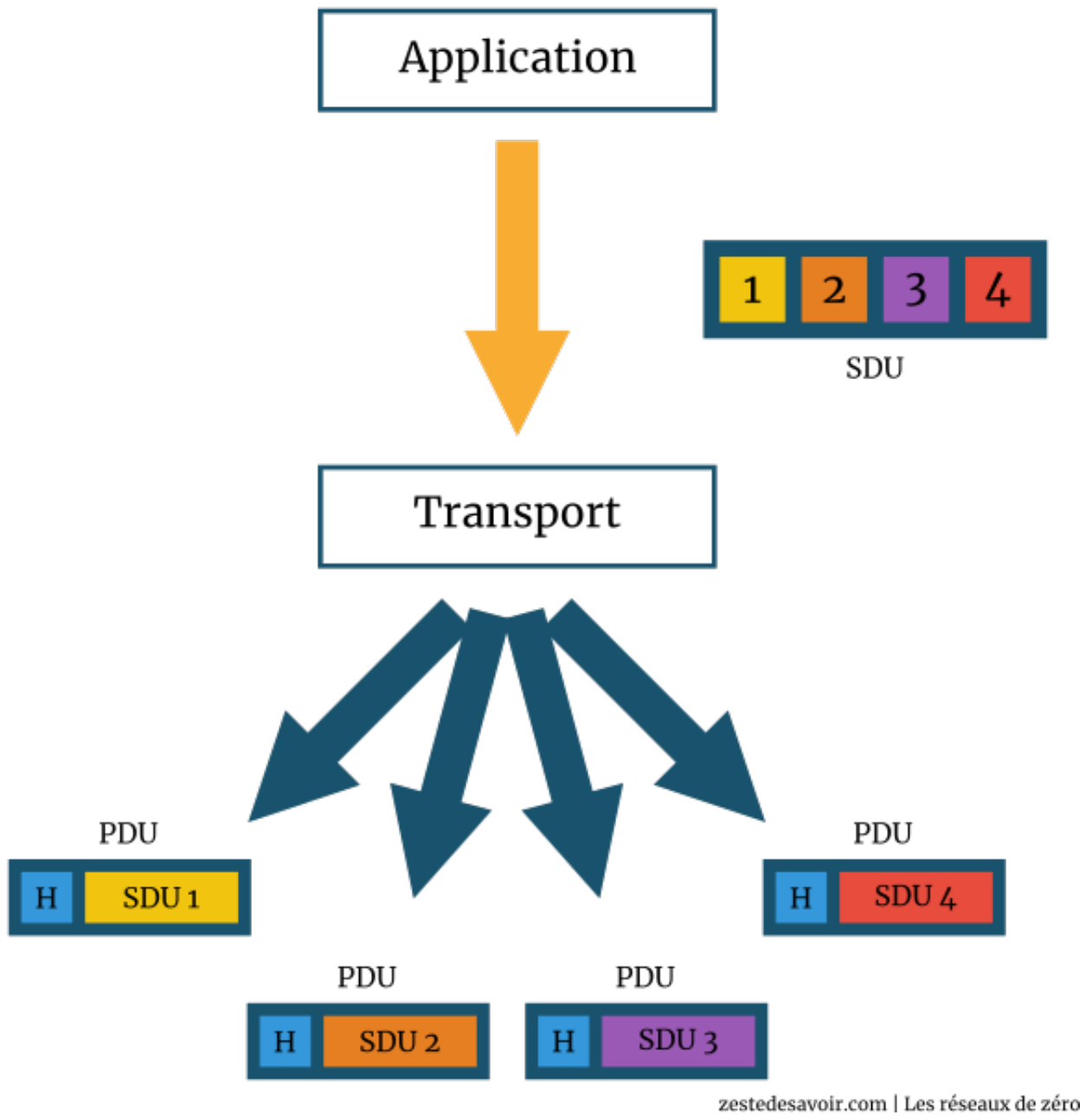


FIGURE IV.1.2. – La couche transport reçoit le SDU et le découpe en 4 (CC BY)

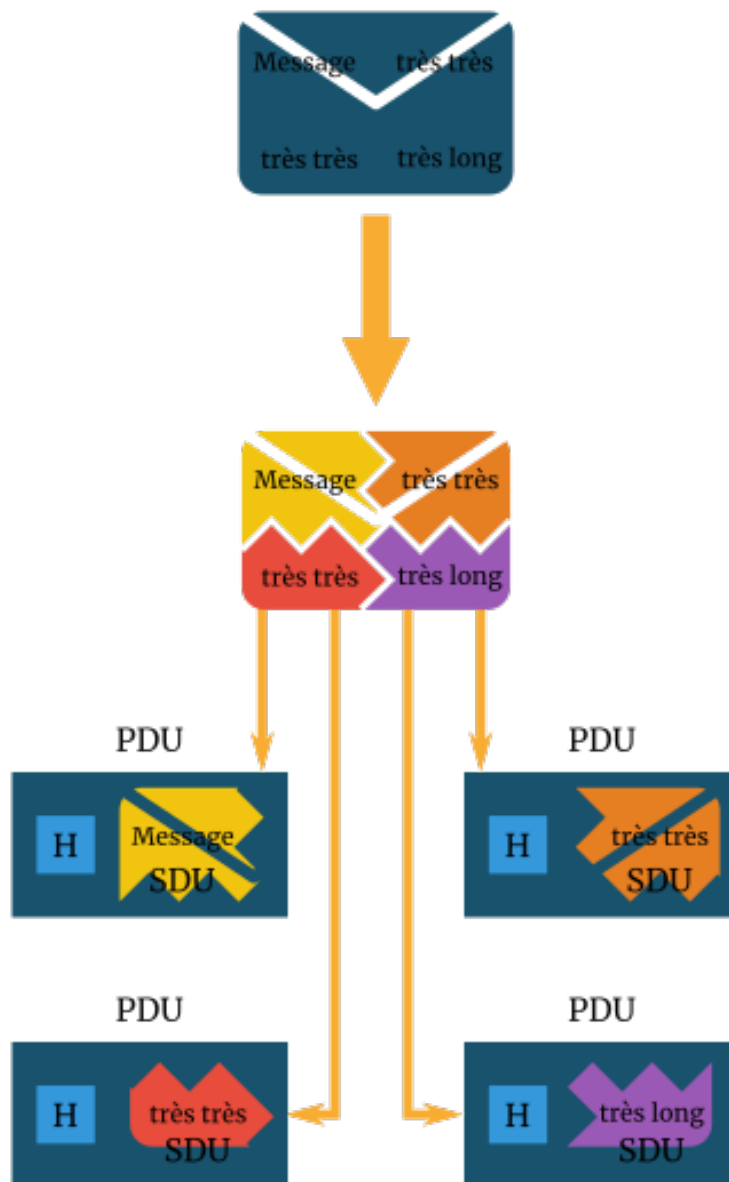


FIGURE IV.1.3. – La couche transport reçoit un e-mail et le découpe en 4 (CC BY)

Ces 4 PDU seront envoyés à la couche C-1 (la couche réseau en l'occurrence). Nous allons sauter ce qui se passe au niveau des 3 dernières couches pour l'instant, car nous ne les avons pas encore étudiées en détail. Mais vous devez avoir une idée générale de ce qui se passe grâce au survol des couches du modèle OSI que nous avons effectué dans l'introduction aux modèles de communication.

Du côté de l'hôte récepteur B, voici ce qui va se passer. La couche transport de cet hôte recevra les 4 PDU originaux, lira leurs en-têtes et les rassemblera en une seule unité de donnée, après avoir supprimé les en-têtes bien sûr (n'oubliez pas les règles du principe de l'encapsulation 🍊). Finalement, le « message » sera transmis au processus de l'application réceptrice.

?

C'est quoi un processus ?

#### IV. Dans les basses couches du modèle OSI

On parle plus de processus en programmation qu'en réseau, mais bon, il est important que vous sachiez ce que c'est. Un processus est une instance d'une application en cours d'exécution (l'application qui est **actuellement** exécutée). Parfois, un processus peut être constitué de plusieurs *threads* (des « sous-processus », en quelque sorte) qui exécutent des instructions au même moment.

C'est tout ce que vous avez besoin de savoir pour l'instant.

##### IV.1.1.2. La relation entre la couche transport et la couche réseau

Vous n'êtes pas sans savoir que, selon le principe des modèles de communication, chaque couche communique avec une couche adjacente (soit supérieure, soit inférieure). Pour vous rafraîchir la mémoire, voici le schéma que nous avons utilisé pour illustrer ce principe de communication entre couches :

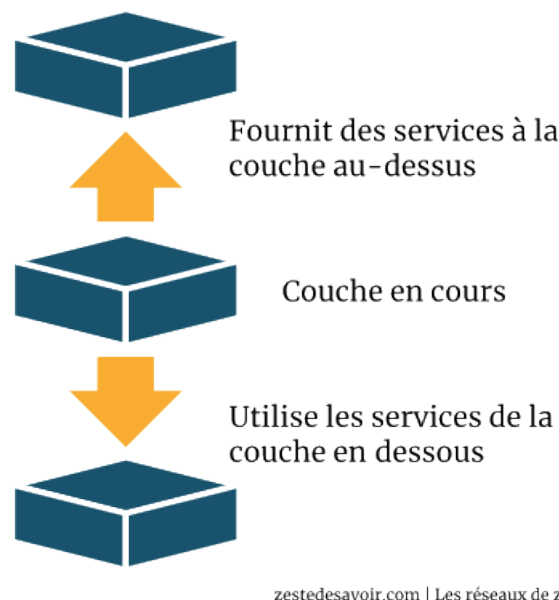


FIGURE IV.1.4. – Représentation schématique d'un modèle en couches (CC BY)

Il se trouve que la couche transport est juste avant la couche réseau dans l'ordre décroissant des couches (de haut en bas, de la couche 7 à la couche 1, de l'application à la couche physique). Il va donc sans dire que la couche transport communique avec la couche réseau.

Nous allons donc voir la relation qui les lie.

Pour commencer, nous avons vu que les deux couches permettent d'établir une communication logique. La couche transport établit une communication logique entre les processus des applications, tandis que la couche réseau établit une communication logique entre les hôtes. Nous allons une fois de plus étudier un scénario. Nous allons imaginer que, dans notre scénario, il y a deux maisons géographiquement opposées : l'une à l'est et l'autre à l'ouest. Chaque maison héberge une douzaine d'enfants. Dans chacune des maisons vivent des familles, de telle sorte que les enfants de la maison à l'est (appelons-la Maison-Est, originalité quand tu nous tiens... 🍊) et ceux de la maison à l'ouest (Maison-Ouest) soient cousins.

#### IV. Dans les basses couches du modèle OSI

Tout est clair jusqu'alors ? Nous ne voulons pas perdre quelqu'un en chemin. 🍊

Chacun des enfants d'une maison écrit à chacun des enfants de l'autre maison, chaque semaine. Chaque lettre est mise dans une enveloppe. Ainsi, pour 12 lettres, nous avons 12 enveloppes. Les lettres sont envoyées par le biais de la poste. Nous avons donc un **taux de transmission** de 288 lettres par semaine (12 enfants x 12 lettres x 2 maisons). Dans la Ville-Est (celle dans laquelle se trouve Maison-Est), il y a un jeune du nom de Pierre (vous devez avoir l'habitude 🍊). Dans la Ville-Ouest (celle dans laquelle se trouve Maison-Ouest), il y a un jeune homme du nom de Jean.

Pierre et Jean sont tous deux responsables de la collecte des lettres et de leur distribution dans leurs maisons respectives : Pierre pour Maison-Est et Jean pour Maison-Ouest.

Chaque semaine, Pierre et Jean rendent visite à leurs frères et sœurs, collectent les lettres et les transmettent à la poste. Quand les lettres arrivent dans la Ville-Est, Pierre se charge d'aller les chercher et les distribuer à ses frères et sœurs. Jean fait également le même travail à l'ouest.

Vous n'êtes pas largués ? Par prudence, voici un schéma illustrant les étapes d'échanges entre les deux familles :

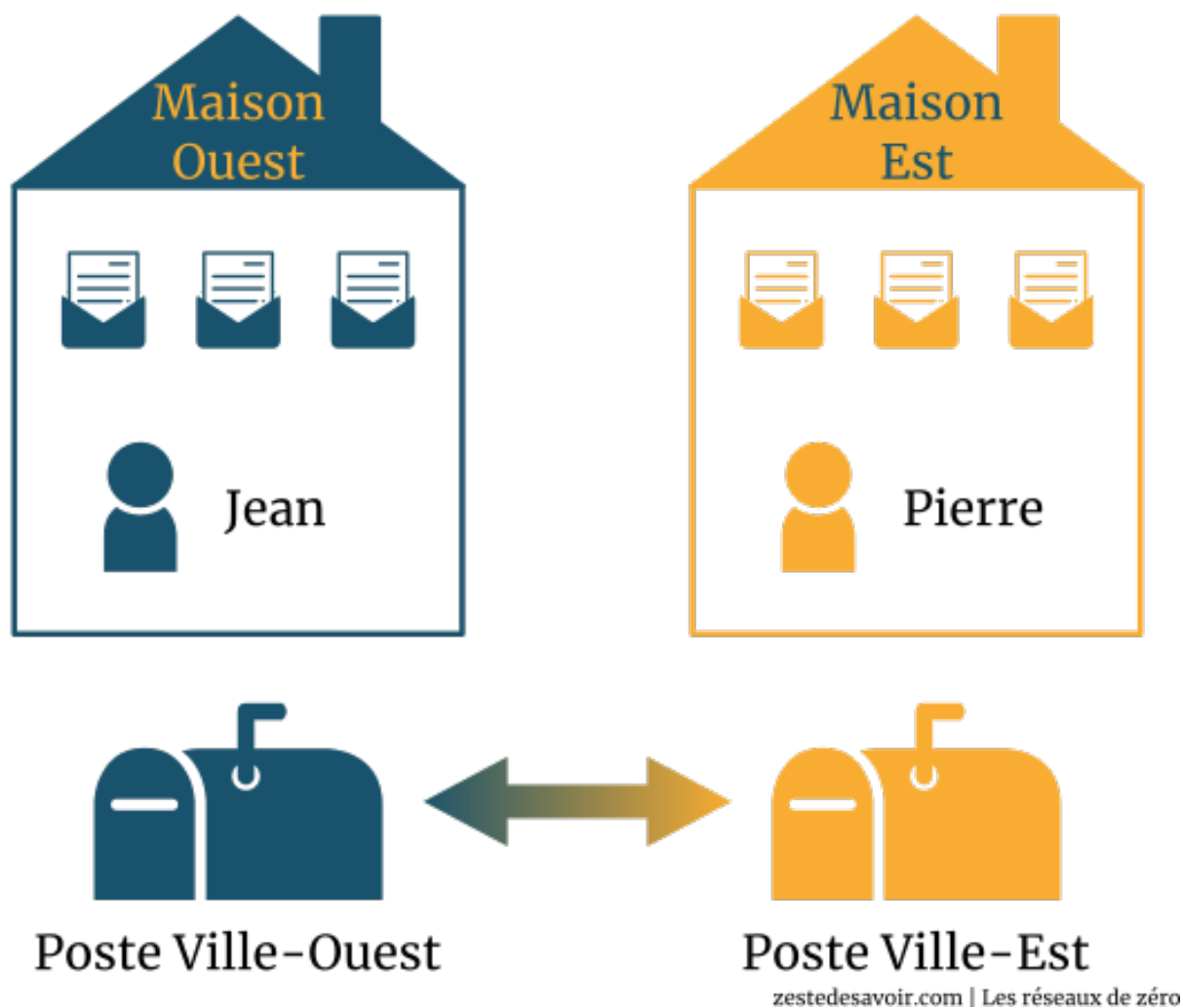


FIGURE IV.1.5. – Schéma représentant les échanges entre les fratries de Jean et Pierre (CC BY)

#### IV. Dans les basses couches du modèle OSI

Le schéma est assez explicite, pas besoin de nous y attarder. Les cousins de Maison-Est écrivent des lettres, Pierre (le grand frère) les collecte et les envoie à la poste de sa ville. La poste de la Ville-Est enverra les lettres à la poste de la Ville-Ouest. Jean va les chercher à la poste et les distribue aux cousins de la Maison-Ouest.

Dans le schéma, nous avons fait en sorte que ces « blocs » soient dans un ordre qui explicite la procédure de transmission. Mais, en fait, pour mieux comprendre cela, il faut suivre la structure semblable au modèle OSI ou TCP-IP, donc une structure de pile (couches superposées). Ceci étant, nous allons réarranger ces blocs ou étapes (qui sont en fait des couches) de manière à retrouver la structure du modèle OSI. Cela donne ceci :

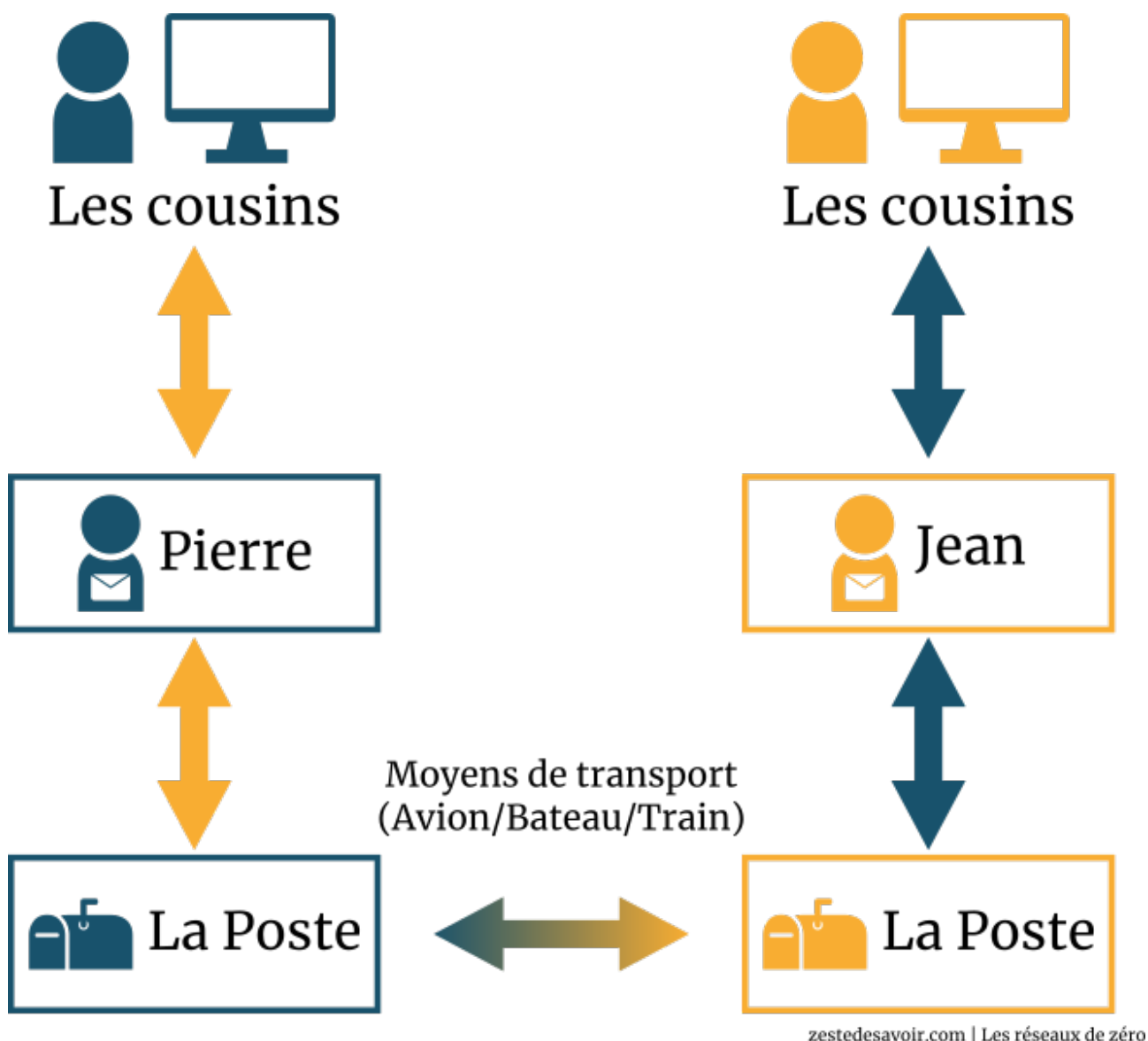


FIGURE IV.1.6. – Structure en pile des échanges entre cousins (CC BY)

Comme vous pouvez le voir sur le schéma, nous avons mis « Moyens de transport » entre les deux postes. Les deux postes ne sont pas dans la même ville, alors comment les lettres seront-elles envoyées ? Par le train ? Par avion ? Il y a plusieurs moyens disponibles. Retenez bien cela !

#### IV. Dans les basses couches du modèle OSI

Maintenant, remplaçons X par sa valeur. 🍊

Dans cet exemple, le bureau de poste pourvoit une **connexion logique entre les deux maisons**. Le service du bureau de poste transporte les lettres de maison en maison et non de personne en personne directement, car il y a un intermédiaire.

Pierre et Jean établissent une **communication logique entre les cousins** (les enfants de Maison-Est et les enfants de Maison-Ouest). Ils collectent et distribuent les lettres à leurs frères et sœurs. Si nous nous mettons dans la perspective (vue, vision, conception) des cousins, Pierre et Jean jouent le rôle du bureau de poste, bien qu'ils ne soient que des contributeurs. Ils font partie du système de transmission des lettres ; plus précisément, ils sont le **bout** de chaque système.

Que vous le croyiez ou non, cette analogie explique très bien la relation qu'il y a entre la couche réseau et la couche transport, ou plutôt l'inverse pour respecter l'ordre. 🍊

- Les maisons, dans notre exemple, représentent les hôtes dans un réseau.
- Les cousins qui vivent dans les maisons représentent les processus des applications.
- Pierre et Jean représentent la couche transport (un protocole de cette couche). Ce sera soit UDP, soit TCP.
- Le bureau de poste représente la couche réseau des modèles de communication (plus précisément, un protocole de cette couche). Le plus souvent, c'est le protocole IP qui est utilisé.

Un schéma n'est jamais de trop. 🍊

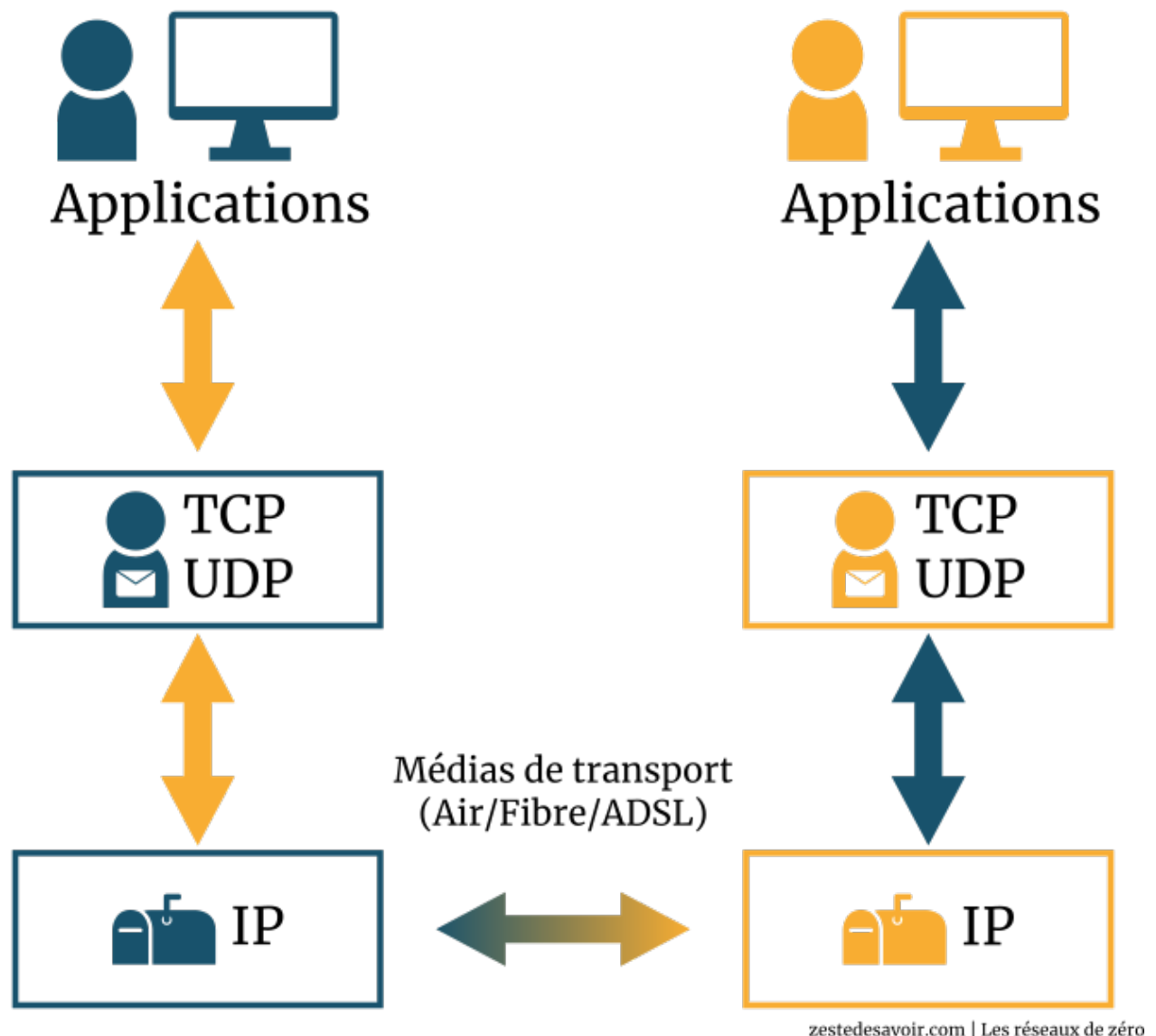


FIGURE IV.1.7. – Transposition des échanges entre cousins avec le modèle OSI (CC BY)

Ainsi, nous pouvons remarquer que le début de la communication (l'écriture des lettres) à l'est commence par des **processus d'une application (les cousins)**. C'est normal, nous sommes dans la couche application. Ensuite, nous nous retrouvons avec plusieurs **PDU** (les enveloppes) qu'un protocole (TCP ou UDP) de la couche transport (Pierre) prendra et transmettra à un protocole de la couche réseau (le bureau de poste).

Dans la procédure de réception, un protocole de la couche réseau (le bureau de poste) dans le réseau (la ville) de l'hôte récepteur (Maison-Ouest) recevra les PDU (enveloppes) qui proviennent d'un protocole de la couche réseau (le bureau de poste) du réseau (la ville) de l'hôte émetteur.

Tout est clair ? Continuons. 🍊

Ce protocole va donc lire les en-têtes et donner les PDU à un protocole de transport (Jean). Ce dernier va transporter les données à l'hôte récepteur (Maison-Ouest) et les distribuer aux processus d'applications (cousins).

#### IV. Dans les basses couches du modèle OSI

Voilà, nous avons remplacé X par sa valeur. Mais ce n'est pas fini, gardez vos ceintures attachées et continuons. 🍊

Comme vous pouvez le comprendre à partir de cette analogie, Pierre et Jean ne sont impliqués que dans des tâches précises qui concernent leurs maisons respectives. Par exemple, ils distribuent les lettres et les transmettent à la poste. **Mais ils n'ont rien à voir avec les activités propres à la poste, telles que le tri des lettres.** Il en est de même pour les protocoles de transport. Ils sont limités au « bout » des échanges. Ils transportent les messages (enveloppes) d'un processus à la couche réseau et de la couche réseau à un processus. Ce sont des fonctions « **bout-en-bout** » (*end-to-end*).

Personne n'est perdu ? Ça serait embêtant de faire demi-tour pour chercher ceux qui se sont paumés. 🍊

Continuons !

Nous allons supposer qu'un jour, Pierre et Jean partent en vacances. Ils se feront donc remplacer par une paire de cousins, disons Jacques et André. Étant donné que ces deux derniers sont nouveaux, ils n'ont donc pas assez d'expérience dans la gestion et la distribution des lettres. Par conséquent, il peut arriver que, de temps en temps, ils fassent tomber des enveloppes en les emmenant à la poste ou en allant les chercher. Donc, de temps en temps, **il y a des pertes d'informations (lettres)**. Puisqu'ils ont du mal à s'habituer, ils seront plus lents que Pierre et Jean. Donc la transmission des lettres à la poste et leur distribution aux cousins prendront un peu plus de temps.

Il se trouve donc que la paire Jacques et André n'offre pas **le même modèle de service** que la paire Pierre et Jean. 🍊

Par analogie, dans un réseau informatique, il est tout à fait possible qu'il y ait plusieurs protocoles de transport. D'ailleurs, vous savez d'ores et déjà que les plus utilisés sont TCP et UDP. Ces protocoles de transport offrent des modèles de services différents entre applications. Il est important de noter que la qualité des services offerts par Pierre et Jean dépend étroitement de la qualité de service offerte par le bureau de poste. C'est logique, en fait !

Si la poste met trois jours, par exemple, pour trier les courriers reçus, est-ce que Pierre et Jean peuvent être tenus responsables de ce délai ? Peuvent-ils transmettre les lettres aux cousins plus vite que ne peut trier la poste ? Non, bien entendu. 🍊

De même, dans un réseau informatique, la qualité de service du protocole de transport dépend de la qualité de service du protocole de réseau. Si le protocole de la couche réseau ne peut garantir le respect des délais pour les PDU envoyés entre les hôtes, alors il est impossible pour les protocoles de transport de garantir le respect des délais de transmission des unités de données (messages) transmises entre applications (processus d'applications).

!

La qualité de service offert par Pierre et Jean dépend étroitement de la qualité des services offerts par la poste, mais l'inverse n'est pas vrai. C'est logique.

?

Un exemple ?

#### IV. Dans les basses couches du modèle OSI

Pas de problème ! Jean, par exemple, peut garantir à ses cousins que, coûte que coûte, « vos lettres seront transmises ». Mais la poste peut perdre des lettres lors du tri, par exemple. 🍊 Jean a offert un service **fiable** (*reliable* en anglais 🍊 ) alors que la poste a offert un service **non fiable** (*unreliable*). Un autre exemple est que Jean peut garantir à ses frères qu'il ne lira pas leurs lettres. Mais qui sait ce qui se passera à la poste ? Peut-être qu'un type méchant pourrait ouvrir une enveloppe, lire la lettre, la refermer et la transmettre ! 🦊 Pire encore, au niveau de la poste, le monsieur peut lire, **faire une photocopie** et envoyer l'original mais garder une copie. Il a brisé la confidentialité de la transmission. Ceci est un concept **très important** en sécurité. Un protocole de transmission peut offrir des services de chiffrement pour protéger et garantir la confidentialité des unités de données mais ces dernières peuvent être **interceptées** au niveau de la couche réseau par [l'attaque de l'homme du milieu](#) 🗑️ (MITM, *Man In The Middle attack*), par exemple.

##### IV.1.1.3. Conclusion

Cette première sous-partie est un vrai baptême du feu en contenu, il faut l'avouer. 🍊 Nous avons vu à quoi servait la couche transport. Ce n'est pas fini ! Ce que nous vous avons montré n'est qu'une présentation introductive. L'exploration de la couche de transport se fera en plusieurs chapitres, donc il y a plein de choses à apprendre. Nous avons simplement vu la différence entre la communication logique offerte par la couche de transport et celle offerte par la couche de réseau. Nous avons profité de notre super exemple ( 🍊 ) pour introduire subtilement le principe d'une communication fiable et non fiable. Vous l'ignorez peut-être, mais dans cet exemple, il y a de l'UDP *inside*. 🍊 Nous y reviendrons.

Ce qu'il faut retenir est que la couche transport établit une communication logique entre processus tandis que la couche réseau établit une communication logique entre hôtes.

##### IV.1.2. À votre service

Continuons notre exploration en nous intéressant à ce que la couche 4 peut faire. Nous avons vu que le rôle de chaque couche était d'offrir des services pour les couches adjacentes. La couche transport n'échappe pas à la règle. Cette couche est un ensemble de services. Ces services sont généralement fournis par un protocole ou une bibliothèque externe. Explorons les concepts les plus importants des services offerts par la couche 4. Nous sommes toujours dans un chapitre d'introduction, il ne s'agit là que d'un passage en revue ! 🍊

- **Communication orientée connexion** : de l'anglais *connection-oriented communication*, ce service est bénéfique pour plusieurs applications, notamment parce qu'il interprète la connexion établie entre processus comme étant un flux de données. Ce flux de données n'est en fait qu'une séquence d'octets (8 *bits*). Ainsi, il faudrait donc gérer l'orientation des octets, qui est également un service de cette couche. Il est plus facile de gérer ce genre de communication que d'avoir affaire au mode orienté non connexion.
- **L'ordre de livraison** : nous savons que la couche transport fragmente les données en plusieurs séquences. La couche réseau ne peut pas garantir que les données envoyées arriveront exactement dans l'ordre même de leur envoi. Rappelez-vous, nous avons parlé de cela en introduction aux protocoles, mais sous un autre nom : le contrôle de séquences. 🍊 Eh oui, c'est la couche transport qui se charge de gérer les séquences, de s'assurer

#### IV. Dans les basses couches du modèle OSI

que l'ordre de livraison est le même que l'ordre d'envoi. Cette gestion de séquences se fait en utilisant un numéro de séquence.

- **Fiabilité** : dans la section précédente, dans l'exemple des cousins et postes, nous avons évoqué implicitement ce qui régissait le principe d'une communication fiable. Le but de ce service est de vérifier l'intégrité des paquets car ces derniers peuvent se perdre en cours de transmission à cause de la congestion du réseau. Cette vérification d'intégrité se fait souvent en utilisant un code de détection d'erreurs tel que la **somme de contrôle** (*checksum*). La fiabilité consiste aussi à s'assurer que le récepteur reçoit chaque segment qui lui est envoyé. Il faudrait donc également être en mesure de renvoyer les séquences qui se sont perdues : cela se fait par une **requête automatique de renvoi**.
- **Contrôle de flux** : il peut être important de réguler le débit au sein même d'un flux. Un certain protocole est capable de s'adapter à son environnement réseau pour adapter sa propre vitesse de communication.
- **Contrôle de congestion** : il sert à contrôler le trafic dans un réseau afin de réduire la congestion (saturation) d'un réseau, en évitant que les clients n'utilisent intensivement les liens de connexion qui sont déjà extrêmement utilisés.
- **Multiplexing / demultiplexing** : nous ne pouvons pas vous parler de *multiplexing* et *demultiplexing* sans aborder les ports de communication, ce que nous allons faire dans le chapitre suivant. Patience ! 🍊

Nous allons nous pencher sur ces services dans les prochains chapitres. Mais vu la quantité d'informations contenues dans celui-ci, on va faire une pause, d'abord ! 🍊

## Conclusion

Ne passez pas tout de suite au chapitre suivant ! Prenez d'abord le temps de bien assimiler tout ce que vous avez appris dans ce chapitre.

...

Ça y est ? Alors l'exploration continue ! 🍊

## IV.2. Exploration de la couche transport

### Introduction

Après une longue introduction, l'heure est à l'exploration de la couche transport ! Au menu : ~~rôti de porc~~ découverte des ports, multiplexage et sockets. Bon appétit !

#### IV.2.1. Appelle mon numéro... de port

##### IV.2.1.1. Les numéros de port : késaco ?

Déjà, qu'est-ce qu'une adresse **IP** ? Vous vous en souvenez ? C'est une adresse unique assignée à un hôte pour permettre de distinguer chaque hôte dans des réseaux. Vous vous en souveniez, bien sûr.

Qu'est-ce qu'une adresse physique ( **MAC** ) ? C'est une adresse associée à la carte réseau d'un hôte pour distinguer les hôtes dans un réseau local. Jusque-là, pas de problème.

Ces deux adresses ont-elles quelque chose en commun ? Oui, elles servent toutes les deux à **identifier** un hôte. 🍏

Un numéro de port sert également à identifier *quelque chose*. Mais ce *quelque chose* est une application. Le numéro de port est donc le numéro qui nous permet de faire la distinction entre les applications. Nous savons déjà que la couche transport établit une communication bout-à-bout entre les processus d'applications. Alors comment faire pour distinguer les nombreux processus d'une application qui sont en fait des services exécutés sur une machine ? C'est le numéro de port qui permettra de les différencier.

Par exemple, nous avons vu que, dans la transmission d'un mail, le premier service ou la première application utilisée est un MUA. Le MUA utilise le protocole **SMTP** pour envoyer le mail au serveur de messagerie souvent en passant par un MSA. Dans l'ordre de réception, on utilisait également un autre MUA pour retirer le mail du serveur de messagerie avec un protocole de réception comme IMAP ou POP. Il se peut donc que les deux services (POP/IMAP et **SMTP**) soient exécutés au même moment sur une même machine hôte. C'est là qu'intervient le numéro de port, parce qu'il nous permet de faire la distinction entre les services qui ont été demandés par l'application distante, qu'il s'agisse d'un serveur de messagerie ou d'un client de messagerie. Le protocole **SMTP** utilise le protocole **TCP** pour la transmission au numéro de port 25.

L'organisme **IANA** (*Internet Assigned Numbers Authority*) [\(en\)](#)  classe les numéros de port en trois catégories principales, comme l'illustre le tableau ci-dessous.

#### IV. Dans les basses couches du modèle OSI

Portée	Catégorie	Description
0 - 1023	Ports bien connus	Ports réservés connus (web, e...
1024 – 49 151	Ports réservés	Ports réservés applications prop...
49 152 – 65 635	Ports dynamiques	Ports « libres pour vos appli des services bie entreprise quel

Voici quelques ports bien connus :

Protocole	Description	Protocole de transmission
<i>File Transfert Protocol</i> (FTP)	Protocole de transfert de fichier	TCP
<i>Secured SHell</i> (SSH)	Protocole permettant l'échange de données par le biais d'un canal sécurisé	TCP & UDP
<i>Telnet</i>	Utilisé pour l'établissement des sessions à distance	TCP
<i>Simple Mail Transfer Protocol</i> (SMTP)	Protocole d'envoi de courrier électronique	TCP
<i>WHOIS protocol</i>	Protocole ou service utilisé pour l'identification d'une machine par son nom de domaine ou son adresse IP. La procédure d'identification se fait par une requête envoyée à un des registres Internet pour obtenir des informations.	TCP
<i>Domain Name System</i> (DNS)	Protocole de résolution des noms de domaine	TCP & UDP
<i>HyperText Transfer Protocol</i> (HTTP)	Protocole de téléchargement (principalement de pages web)	TCP & UDP
<i>Post Office Protocol Version 2</i> (POP2)	Protocole de retrait de mails d'un serveur de messagerie	TCP
<i>Post Office Protocol Version 3</i> (POP3)	Protocole de retrait de mails d'un serveur de messagerie	TCP
<i>Internet Message Access Protocol</i> (IMAP)	Protocole de retrait et consultation de mails d'un serveur de messagerie	TCP & UDP
...	...	...

Il y a tellement de protocoles que nous ne pouvons pas tous les énumérer. [Voici une liste des numéros de port](#) <sup>(en)</sup> ↗ .

## IV. Dans les basses couches du modèle OSI

Expliquons quelques en-têtes de ce tableau avant de clore cette section.

- **Statut d'assignation** : le statut officiel veut dire que le couple application / numéro de port a été enregistré dans les registres de l' **IANA**. En d'autres termes, il est défini par une convention que telle application utilise tel numéro de port. Il y a également des applications qui utilisent des numéros de port non officiels, mais nous ne les avons pas listées. Vous pourrez les trouver en suivant le lien que nous vous avons donné.
- **Protocole de transmission** : vous devez normalement savoir que les protocoles peuvent s'utiliser entre eux. Le protocole **SMTP** qui sert à envoyer un mail s'appuie sur le protocole **TCP** pour le transmettre. On parle de sous-couchage de protocoles (*underlayering protocols*). Vous pouvez voir que certains protocoles n'utilisent que **TCP** ou **UDP** alors que d'autres peuvent utiliser les deux, c'est au choix.

### IV.2.1.1.1. Les notions autour du numéro de port

Les ports sont un concept très important en réseau. Grâce à ces derniers, nous pouvons faire beaucoup de choses très utiles et intéressantes. Nous allons voir quelques-unes de ces notions qui gravitent autour des ports. Pour l'instant, nous allons vous en fournir une brève description pour la culture. Plus de détails dans la (future) partie du cours consacrée aux services. 🍌

### IV.2.1.1.2. La redirection de port (port forwarding)

Le *port forwarding* ou encore *port mapping* permet à un hôte d'un réseau distant (Internet par exemple) de communiquer avec un hôte d'un réseau local en faisant transiter les paquets par une machine tel qu'un routeur. L'intérêt est que l'hôte local n'est pas directement sollicité par d'autres, une machine fait le relai et peut ainsi servir de filtre.

### IV.2.1.1.3. Le scan de port (port scanning)

Le scan ou balayage de port est une technique très populaire en réseau et surtout en sécurité. Cette technique consiste à « scanner » un hôte afin de découvrir les ports ouverts et d'avoir la liste des services exécutés sur cet hôte. Connaître les ports ouverts et les services exécutés permet à un pirate, par exemple, d'exploiter les vulnérabilités des services afin de planifier une attaque. 🦉

### IV.2.1.1.4. Déclenchement de port (port triggering)

Le *port triggering* consiste, comme son nom l'indique, à déclencher quelque chose. Cette technique permet de déclencher l'ouverture d'un port précis. Le *port triggering* ne se produit donc que lorsqu'un événement déclencheur particulier a lieu. Par exemple, l'ouverture d'une connexion sur le port 60000 va déclencher une redirection de port.

#### IV.2.1.5. PAT : Port Address Translation

**PAT** permet à plusieurs hôtes dans un réseau local d'utiliser une même adresse **IP** d'un réseau public. On l'utilise dans le même but que le *subnetting*, c'est-à-dire pour mieux gérer les adresses disponibles. **PAT** est une technique qui opère dans les couches 3 et 4 du modèle OSI. L'un des défauts de cette technique est la complexité que cela implique au niveau du pare-feu ou du routeur qui l'implémente : il doit jongler avec les numéros de ports pour suivre les communications en cours et éviter de confondre plusieurs hôtes.

#### IV.2.2. Multiplexing / demultiplexing

Dans le chapitre précédent, nous vous avons dit que nous ne pouvions pas parler du *multiplexing* / *demultiplexing* (en français : multiplexage / démultiplexage) sans aborder la notion des ports. Maintenant que c'est fait, nous pouvons parler ~~du beau temps, du soleil et du vent~~ de ces deux notions fondamentales. 🍊 Dans l'exemple des cousins, frères, etc., nous avons implicitement parlé du *multiplexing* et *demultiplexing*. Nous avons également parlé des principes de transmissions fiables et non fiables ( **TCP** et **UDP**). Tout est dans l'exemple. Les plus caïds d'entre vous seront capables de remplacer X par sa valeur, comme nous l'avons fait.

L'hôte-récepteur, lors d'un échange de données, reçoit les PDU de la couche réseau. Nous avons vu que le rôle de la couche transport est d'acheminer ou de donner les PDU reçus à qui de droit, c'est-à-dire aux processus d'application. Ces applications sont identifiables par un numéro de port, comme nous l'avons vu. Il se peut que plusieurs services (processus d'applications) soient exécutés au même moment (ce qui est presque tout le temps le cas). Certaines applications peuvent avoir plusieurs instances en cours d'exécution. Par exemple, vous pouvez utiliser Firefox et Chrome au même moment. Ces deux applications vous donnent accès aux services du protocole HTTP. Supposons alors que vous ayez deux processus d'application HTTP, et deux sessions à distance (Telnet) en cours d'exécution sur une machine. Alors, lorsque la couche transport aura reçu les PDU de la couche réseau, elle va examiner les en-têtes de ces PDU afin de retrouver l'identifiant du processus auquel le PDU doit être acheminé. **C'est cela, le démultiplexage.** C'est le fait de transmettre un PDU donné au processus d'une application donnée.

Si vous vous souvenez de notre exemple des cousins, Jean et Pierre sont responsables du démultiplexage, étant donné que lorsqu'ils reçoivent les enveloppes du bureau de poste, ils doivent vérifier à qui est destinée chaque enveloppe, et finalement la remettre à qui de droit. Il y a 12 enfants dans chaque maison, avons-nous dit. Ce tri et l'acheminement d'une lettre à la bonne personne, c'est cela le démultiplexage en réseau. 🍊

Le multiplexage est exactement le contraire du démultiplexage. Ça devrait donc vous paraître évident. Si, à la réception (avec Jean), la distribution de chaque lettre à son destinataire est le démultiplexage, en toute logique le multiplexage sera donc « l'encapsulation » des lettres et leur acheminement à la poste locale (avec Pierre, donc).

Quand les cousins de Maison-Est écrivent des lettres, Pierre les collecte et met chaque lettre dans une enveloppe. Sur l'enveloppe, il met le nom de l'émetteur et le nom du destinataire. Ensuite, il donne ces enveloppes au bureau de poste. Sa mission **s'arrête là**. Ainsi, un protocole de la couche transport est responsable de la collection des SDU, de leur encapsulation, en spécifiant le numéro de port du processus de l'application utilisée et le numéro de port à utiliser pour le processus de l'application réceptrice. Cette encapsulation, c'est la transformation du SDU en

#### IV. Dans les basses couches du modèle OSI

PDU. Ce protocole est aussi responsable de la livraison de ce PDU à un protocole de la couche inférieure (couche réseau, protocole **IP**).

Schématiquement, ça donne ceci :

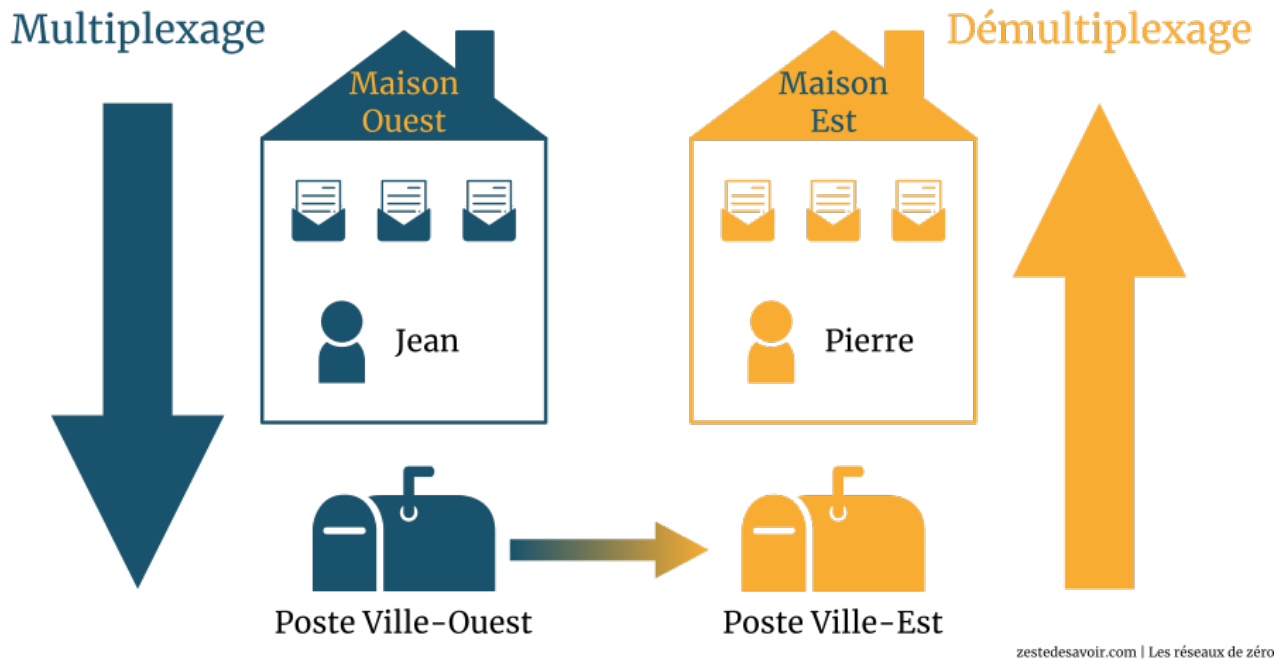


FIGURE IV.2.1. – Multiplexage / démultiplexage (CC BY)

Le schéma est tellement clair ! Les processus (cousins) écrivent des lettres (SDU). Un protocole de transport (Pierre), qui peut être **TCP** ou **UDP**, accomplit une opération de multiplexage en rangeant chaque SDU dans son enveloppe (c'est l'encapsulation). Nous nous retrouvons donc avec 12 PDU (12 cousins ont écrit des lettres). Le protocole **TCP** ou **UDP** transmet ces PDU au protocole réseau **IP** (le bureau de poste). Le protocole **IP**, bien entendu, fera plusieurs autres choses que nous (protocole de transport) n'avons pas besoin de connaître. Quand vous allez déposer un courrier à la poste, le reste, ce n'est plus votre affaire : vous avez fait votre part, à la poste d'assumer ses responsabilités. 🍌 Une fois que ces lettres ont emprunté un média de transmission (câble, etc.), elles vont arriver au niveau de la couche réseau de l'hôte récepteur. Là, également, le protocole réseau **IP** va accomplir un certain nombre d'opérations qui ne nous intéressent pas. Après avoir terminé son travail, ce dernier va faire appel à un protocole de transport (Jean), lequel effectuera une opération de démultiplexage, en transmettant chaque enveloppe à son destinataire.

Les protocoles **TCP** et **UDP** sont donc **responsables de la modification des en-têtes des unités de données lors du multiplexage / démultiplexage**.

##### IV.2.2.1. Structure partielle de l'en-tête de transport

Comme nous l'avons dit plus haut, le multiplexage et démultiplexage sont des opérations effectuées par un protocole de transport. Ces opérations modifient les en-têtes des unités de données. Nous parlons de structure partielle parce que les valeurs que nous allons examiner ne sont pas les seules qu'il y a dans les en-têtes d'un protocole de la couche 4. Il y a plusieurs

autres choses que nous allons progressivement découvrir. Pour l’instant, contentons-nous de voir la structure partielle d’un segment :

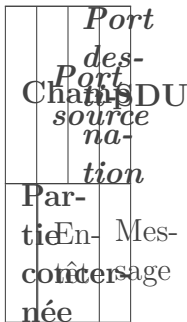


TABLE IV.2.4. – Structure basique d’un protocole de niveau 4

Comme vous le voyez, un segment de protocole de transport est partiellement constitué des champs « port source » (*source port*), « port de destination » (*destination port*) et « SDU ». Les parties en italique sont des champs membres de l’en-tête. Vous êtes censés savoir ce qu’est le SDU : nous en avons parlé dans la sous-partie traitant du principe d’encapsulation. Il n’y a pas grand-chose à dire sur ces champs, tout est déjà si clair. Le champ « port source » contiendra le numéro de port utilisé par l’application émettrice. Le champ « port de destination » contiendra le numéro de port identifiant l’application réceptrice. Finalement, le champ « SDU », c’est le message original. Alors, si l’application utilisée était un service de résolution de nom de domaine (DNS), et que le contenu du message était « 192.168.0.1 -> zestedesavoir.com », à quoi ressemblerait partiellement le segment généré par un protocole de transport ? Vous êtes en mesure de résoudre cet exercice tout seul. Vous pouvez vous référer à la liste des numéros de ports bien connus. Nous allons *supposer* que l’application source, comme l’application réceptrice, utilise le même numéro de port.

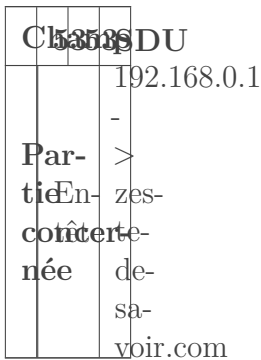


TABLE IV.2.6. – Contenu supposé de l’échange DNS au niveau 4

?

Pourquoi un numéro de *port source* et un numéro de *port destination* ?

Vous savez tous ce qu’est un port et un numéro de port. Un numéro de port est un nombre entier codé sur 16 *bits* (2 octets). Ainsi, il peut avoir une valeur allant de 0 à 65 535. Nous savons que chaque type d’application a un numéro de port, alors vous vous demandez « Comment ça se fait que la structure de notre segment de transport contienne deux numéros de port ? Un seul devrait suffire, non ? 🍊 ». La réponse est... non. Un seul numéro de port ne suffirait pas !

#### IV. Dans les basses couches du modèle OSI

Supposons que l'hôte A ait une instance d'une application HTTP en cours d'exécution (disons qu'il surfe sur le web) et qu'il envoie une requête (il demande une page) à un serveur web B qui a plusieurs instances du service HTTP en cours d'exécution. Chaque instance a le même numéro de port, soit 80, selon les numéros de ports bien connus.

i

Les serveurs web créent un nouveau processus pour chaque requête HTTP qu'ils reçoivent. Un serveur qui gère 10 requêtes gère donc 10 processus utilisant tous le même numéro de port.

Alors, comment le protocole de transport au niveau du serveur web va réussir son démultiplexage (acheminer la requête au processus adéquat) ? Il y a plusieurs processus exécutés au même moment. En se basant uniquement sur un seul numéro de port dans le segment de transport, le démultiplexage va échouer. 🍊

Nous avons donc besoin de deux numéros de port (source et destination). OK, mais comment savoir quelle valeur mettre dans tel champ du segment ? Dans le mini-exercice que nous vous avons donné plus haut, nous avons dit de **supposer** que les deux hôtes utilisaient un même numéro de port. Ce n'est pas le cas, en réalité. Donc vous rencontrerez un segment de transport avec des valeurs différentes dans les champs « *Source port* » et « *Destination port* ».

Comme vous devriez le savoir, normalement, en réseau, la communication entre deux hôtes se fait selon une architecture client-serveur. Celui qui initialise la transmission est le client, celui qui répond est le serveur. Ainsi, si nous avons un hôte junior0-PC qui essaie de communiquer avec un hôte vince-PC *via* une session à distance (Telnet), quelles seront les valeurs des champs « *Source port* » et « *Destination port* » ? Eh bien, le champ « *Destination port* » aura pour valeur 23, car c'est le numéro de port bien connu, conventionnellement attribué par l'IANA pour Telnet. Mais quid du champ « *Source port* » ? Quelle valeur allons-nous y mettre ? 23 ?

En général, le protocole de cette couche ( **UDP** ou **TCP** ) va générer automatiquement un numéro de port qu'aucun processus n'utilise actuellement. Comme vous le savez, plusieurs processus (parfois plusieurs processus d'un même type d'application) peuvent être exécutés au même moment sur une même machine. Chaque processus a un numéro de port. **On ne peut pas utiliser un numéro de port déjà utilisé par un autre processus pour le champ « *Source port* ».**

?

Ok, cool, mais si je veux spécifier un numéro de port non officiel pour une application que j'ai créée ?

Bonne question ! Si, grâce à la programmation des sockets, vous avez créé une application qui doit utiliser un numéro de port de votre choix, alors, bien entendu, vous pouvez déterminer le numéro de port source à utiliser, et ce dernier sera marqué dans le champ « *Source port* » du segment. Pour ce faire, utilisez la fonction `bind()` de l'API que vous aurez utilisée pour la création de votre application. 🍊

Disons que nous avons programmé une application clemNet ( 🍊 ) de manière à utiliser le numéro de port Y. Nous voulons utiliser clemNet et envoyer une requête HTTP à un serveur web. Nous savons que le serveur écoute (grâce à la fonction `listen()` ) le port 80. Le message que nous envoyons au serveur web sera par exemple l'URL d'un site web : `zestedesavoir.com` (c'est simplifié, une requête HTTP est plus complexe, en réalité).

À quoi ressemblera notre segment de transport ?

				zes-
				te-
				Chaspe-
				sa-
				voir.com
Par-	En-	Mes-		
tie	ête	sage		
conten-				
née				





TABLE IV.2.8. – Contenu supposé de l’échange clemNet au niveau 4

Y sera la valeur du champ « Source port » et 80 la valeur du champ « Destination port ». Le serveur va recevoir ce PDU et examiner sa constitution. « Ah tiens, c’est une requête d’une page web (port 80) par l’application clemNet (port Y) ». Il va traiter la requête et renvoyer une réponse. C’est là qu’on comprend l’importance des deux numéros de port. Dans la procédure d’envoi de la requête, le champ « Source port » avait pour valeur Y. Dans l’envoi de la réponse, le champ « Source port » prend la valeur du champ « Destination port » de la requête, et le champ « Destination port » prend la valeur du champ « Source port » de la requête. Ainsi, notre segment envoyé par le serveur web ressemblera à ceci :

				in-
				dex.html
Par-	En-	Mes-		
tie	ête	sage		
conten-				
née				

TABLE IV.2.10. – Contenu supposé de la réponse HTTP au niveau 4

Pour résumer cela, voici un tableau qui montre les valeurs de chaque champ au niveau du client et au niveau du serveur :

Machine	Port source	Port destination	Ressour
 Clem-PC	Y	80	 /opt/ www.z
 ZdS	80	Y	 /opt/ page d'

Tout est clair ? Fin de la section ! 🍌

### IV.2.3. Introduction aux sockets



Cette section introduit quelques notions relatives à la programmation réseau. Sa lecture n'est pas indispensable pour le reste du cours.

Nous savons maintenant que les protocoles se chevauchent. Un protocole applicatif ( **SMTP**, **POP**, **HTTP**, etc) peut être interfacé à un protocole de transport ( **UDP**, **TCP**). Mais comment faire pour que les processus des applications communiquent avec les ports des protocoles de transport ? C'est à ça que servent les sockets. Un socket est une interface entre les processus : en réseau, un socket sert donc à faire communiquer un processus avec un service qui gère le réseau. Chaque socket a une adresse de socket. Cette adresse est constituée de deux choses : une adresse **IP** et un numéro de port. C'est grâce à la programmation de socket que l'on définit le modèle de communication. Si le socket a été configuré de manière à envoyer ou recevoir, c'est un modèle *half-duplex*. S'il a été configuré de manière à envoyer et recevoir simultanément, il s'agit d'un modèle *full-duplex*. Étant donné que les sockets sont en fait une interface de programmation d'applications (API), on peut donc s'en servir pour programmer des applications en réseaux (par exemple, créer une application pour faire communiquer un client et un serveur). Dans le souci de vous encourager à faire des recherches sur la programmation des sockets, voici un schéma illustrant une communication entre un client et un serveur.

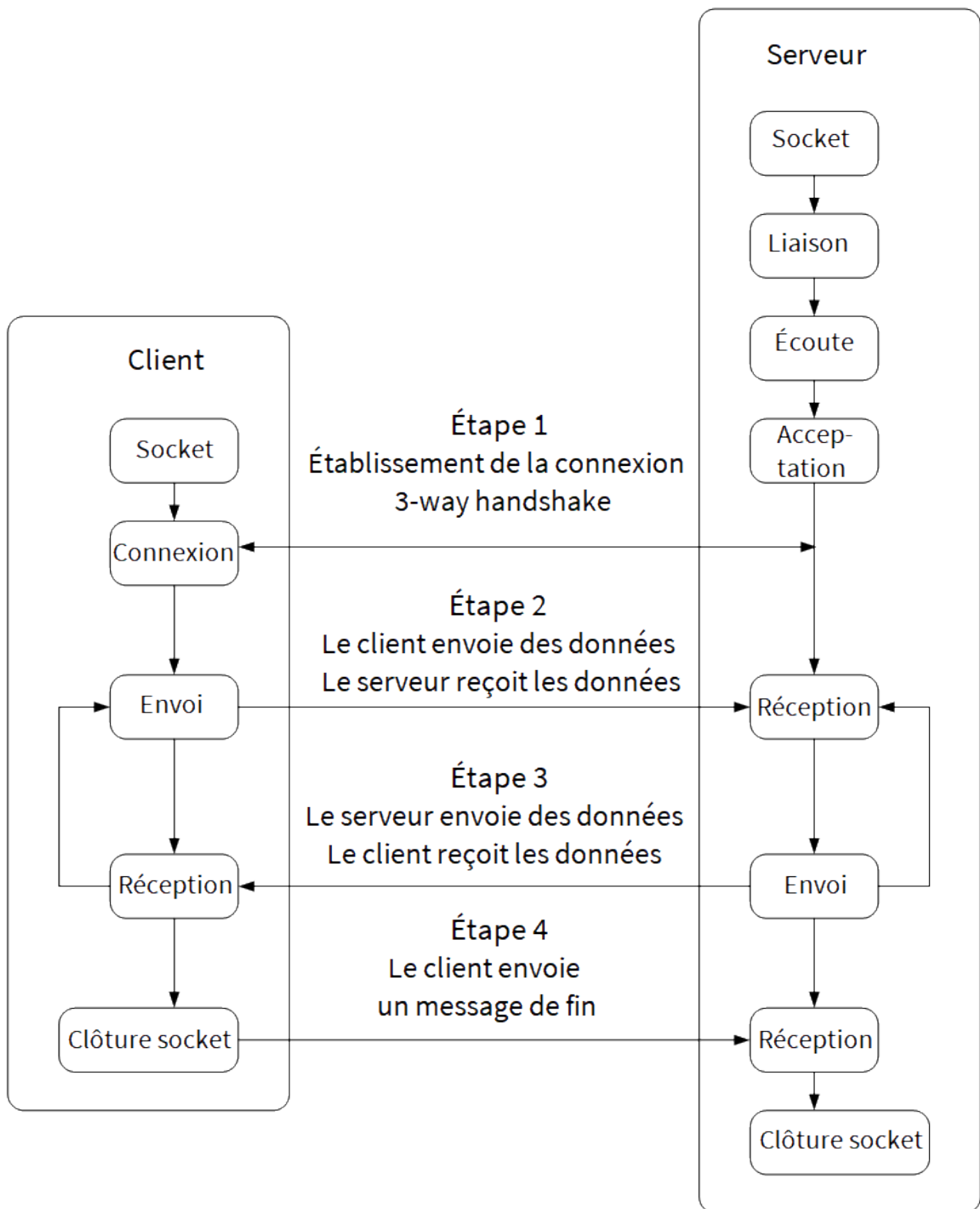


FIGURE IV.2.2. – Schéma représentant le fonctionnement de sockets

Le schéma en lui-même est assez explicite, mais nous allons vous guider afin de bien le comprendre.

Le client commence à se connecter au serveur grâce aux sockets. Une fois la connexion établie (étape 1), le client et le serveur peuvent communiquer (s'échanger des messages). C'est ce qui se passe dans les étapes 2 et 3. À la fin de la communication, le client envoie une demande de

terminaison de session au serveur (étape 4) et le serveur met fin à la connexion. 🍊

Mais avant tout cela, vous remarquerez que le serveur n'effectue pas les mêmes actions initiales que le client. Le serveur utilise les sockets pour lier un port d'application à son processus correspondant. Ensuite, il « écoute » ce port. « Écouter », ici, c'est « continuellement vérifier s'il y a un événement qui se passe sur ce port précis ». Ce faisant, il va découvrir qu'un client essaie de se connecter à lui par le numéro de port qu'il écoute. Il accepte donc la requête, établit la connexion (étape 1) et, finalement, les deux communiquent (étapes 2, 3 et 4). Le modèle de communication est *half-duplex* parce que le client envoie et attend la réponse du serveur et vice-versa.

##### IV.2.3.1. Les fonctions des API



Nous allons un peu entrer dans des expressions propres à la programmation. Si vous n'êtes pas intéressés par la programmation des applications en réseaux, sautez ce petit point et continuez la lecture plus bas.

Comme vous le savez dorénavant, la programmation des sockets se fait par le biais d'une API (Interface de Programmation d'Application). Il existe plusieurs API pour programmer les sockets ; l'une des plus populaires est *Winsock*. Cependant, chaque API propose les fonctions suivantes :

- **socket()** : dans le schéma, vous voyez bien que la première case est « socket ». Cette fonction crée un objet de type Socket que l'on peut identifier par un nombre entier. La création de cet objet, bien entendu, nécessite l'allocation de ressources (mémoire) pour cet objet.
- **bind()** : en français, ça veut dire « lier ». Au niveau du serveur, dans le schéma, nous avons mis « liaison » juste après socket. Ainsi, après avoir créé une nouvelle instance d'un objet de type Socket, au niveau du serveur, il faut utiliser la fonction bind() pour lier ou associer un socket à une adresse de socket ( **IP** + port).
- **listen()** : cette fonction est également utilisée au niveau du serveur. Dans le schéma, c'est le bloc « écoute ». Cette fonction change l'état du socket et le met dans un état d'écoute. Comme nous l'avons expliqué, le serveur va « écouter » l'adresse à laquelle est associé le socket en attendant un événement. Il y a également une fonction poll() qui agit aussi dans le même but que la fonction listen(), mais d'une manière différente.
- **connect()** : cette fonction permet au client d'établir une connexion avec le serveur. En général, ça sera une connexion **TCP**, car la majorité des sockets utilisent **TCP** comme protocole de transmission. Cette fonction assigne également un numéro de port local au socket coté client.
- **accept()** : en toute logique, cette fonction sera appelée du côté du serveur, car elle sert principalement à accepter une requête de connexion envoyée par le client.
- **send()** : cette fonction, qui signifie « envoyer » et qui est également représentée dans le schéma (bloc « envoi »), sert à envoyer des données du client au serveur et vice-versa. On utilise également la fonction write() (écrire) ou sendto() (envoyer à).
- **recv()** : cette fonction, représentée par le bloc « réception » du schéma, sert à recevoir les données envoyées par la fonction send(). On utilise également la fonction read() (lire) ou recvfrom() (recevoir de).
- **close()** : c'est la fonction qui permet au système d'exploitation de libérer les ressources allouées lors de la création de l'objet de type Socket avec la fonction socket(). C'est donc

#### *IV. Dans les basses couches du modèle OSI*

la terminaison de la connexion. Elle est représentée par le bloc « fin » dans notre schéma.

## Conclusion

Nous y sommes enfin ! Ou pas... En effet, nous avons terminé notre tour d'horizon de la couche transport. Néanmoins, nous allons nous attarder sur les deux protocoles capitaux de cette couche 4 : UDP et TCP. Courage ! 🍊

## IV.3. Les protocoles de la couche transport

### Introduction

Nous venons de passer deux chapitres à étudier la couche transport dans le principe. Pour mettre en œuvre tous les services qu'elle offre, elle exploite deux protocoles incontournables qu'il est essentiel de connaître.

Dans ce chapitre, nous allons nous focaliser sur les deux protocoles principaux de cette couche : TCP (*Transmission Control Protocol*) et UDP (*User Datagram Protocol*). Nous commencerons par étudier UDP, qui est assez simple, ensuite nous nous focaliserons sur TCP. Finalement, nous comparerons les deux protocoles afin de déterminer dans quels cas il faut privilégier l'un au détriment de l'autre. TCP étant un protocole richissime, nous allons voir seulement quelques-unes de ses fonctionnalités dans ce chapitre.

#### IV.3.1. UDP, un protocole irresponsable

UDP est l'un des protocoles les plus utilisés au niveau de la couche transport, juste après TCP. Les deux sont des protocoles dont la vocation est de transporter les données d'une application à une autre comme nous l'avons vu. Dans cette sous-partie, nous allons nous concentrer sur UDP.

##### IV.3.1.1. UDP par l'analogie

Pour bien aborder une notion importante, une analogie (vous commencez à être habitués 🍌).

Supposons que vous allez participer à une grande fête chez un ami, Pierre. Votre ami vous a donné la permission d'inviter vos amis, même s'il ne les connaît pas. Vos amis, ne connaissant pas Pierre, ne savent pas comment arriver chez lui. Vous leur avez demandé de vous rejoindre chez vous à 17h, afin de vous rendre à la fête ensemble. Le jour J arrive et tous vos amis sont chez vous. Vous décidez alors d'emprunter des taxis pour aller chez Pierre. En sortant de chez vous, il y a une file de plusieurs taxis disponibles. Vous allez donc vous séparer en plusieurs groupes. Chaque groupe prendra son propre taxi. Le but est le même pour tous : arriver chez Pierre. Mais il est fort probable que d'autres arrivent plus tôt que les autres (à cause de feux de circulation, ...). Il est aussi probable que les taxis n'empruntent pas la même route pour arriver à la destination finale. Pour les malchanceux, il se peut même qu'ils aient un accident en route, les empêchant ainsi d'arriver à destination. Ainsi, les taxis ne peuvent pas garantir que tout le monde arrivera à temps, dans les délais voulus, et surtout que tous vont emprunter le même chemin ou arriver au même moment.

i

Diverses raisons peuvent faire que plusieurs taxis allant d'un point A à un point B ne se suivent pas les uns les autres. Un ralentissement soudain, un accrochage, un contrôle de police peuvent altérer la fluidité de la circulation et faire changer l'itinéraire d'un ou plusieurs véhicules pour arriver plus rapidement à destination.

?

Un autre exemple ?

On décrit souvent UDP comme étant un protocole **fire and forget**, ce qui signifie en français **tire et oublie**. Imaginez que vous entendiez un bruit dans la cour de votre villa. Vous allez dans la cour, mais il fait sombre. Vous remarquez quelques mouvements, et vous prenez votre arme et tirez. Vous **supposez** que ce bandit qui essaie de vous cambrioler est mort et vous rentrez vous recoucher. Vous avez appuyé sur la gâchette et vous êtes rentré sans vous soucier de quoi que ce soit. Est-ce que la balle a atteint la cible ? Ce n'est pas votre problème. Vous avez tiré et après, si la cible est touchée tant mieux, sinon tant pis.

UDP signifie **User Datagram Protocol** (protocole de datagramme utilisateur). C'est un protocole de transport qui ne peut pas garantir que les données arriveront toutes à destination. C'est d'ailleurs l'un des principaux défauts de ce protocole. On ne peut pas être sûr que les données arriveront au destinataire, car UDP n'a pas de mécanisme de retransmission quand un paquet se perd. C'est un protocole **non orienté connexion**.

Dans notre premier exemple, lorsque le taxi a un accident, est-ce qu'il peut soigner ses passagers et ensuite continuer la course vers la destination ? Non ! Donc les passagers n'arriveront pas à destination, en d'autres termes, ils ne seront pas retransmis, donc se perdent. Voilà une petite introduction à ce protocole.

?

Pourquoi « non orienté connexion » ?

Nous avons vu que dans la couche transport, le mode d'envoi était choisi : mode orienté connexion ou non orienté connexion. Pour comprendre le principe de ces deux modes, il nous faut aborder la notion de la **poignée de main** (*handshake*).

### IV.3.1.2. Le principe de la poignée de main



FIGURE IV.3.1. – Illustration d'une poignée de main - domaine public

Pourquoi vous faites cette tête-là ? Ça vous étonne qu'on parle de poignée de main en réseau ? 🍊 Nous ne le dirons jamais assez, la technologie s'inspire du monde réel. 🍊 Alors qu'est-ce qu'une poignée de main ?

Non, c'est pour de vrai, la question est sérieuse. 🍊

Une poignée de main est une action qui se fait en deux mouvements : on tend la main et l'autre personne la prend (elle aussi nous tend la main). On se sert de ce rituel dans plusieurs domaines. Par exemple dans la vie quotidienne, la poignée de main est utilisée lors d'une salutation, alors qu'en *business*, c'est pour signifier une entente, un contrat conclu.

?

Et en réseau ?

En réseau, on se sert du principe de la poignée de main **pour l'initialisation d'une connexion**.

Vous vous souvenez du tout premier chapitre sur les protocoles ? Nous avons vu, par l'exemple d'une conversation au téléphone, comment opéraient les protocoles. La première étape d'une communication était l'établissement de la connexion, n'est-ce pas ?

En effet, lorsque vous téléphonez à un ami, il décroche en disant « Allô » et vous répondez « Allô ». C'est le même principe que la poignée de main, vous vous êtes mis d'accord sur le fait que vous allez engager une conversation.

En réseau, le principe reste fondamentalement le même. Quand vous voulez communiquer avec un autre hôte dans un réseau, il faut que les deux hôtes se mettent d'accord. Cette entente se fait par la poignée de main. En fait, l'hôte émetteur envoie un paquet qui dit « Est-ce qu'on peut parler ? » et l'hôte récepteur répond « Vas-y, j'écoute ». Et là les deux peuvent s'échanger des données ! 🍊

?

Le rapport avec UDP ?

Justement, UDP n'est pas orienté connexion du fait qu'il n'implémente pas ce principe de la poignée de main. En d'autres termes, lorsqu'une application utilise UDP comme protocole,

les données sont envoyées sans au préalable établir une session ou une connexion avec l'hôte récepteur. C'est pour cela que UDP ne garantit pas que les données vont arriver. Quand vous utilisez UDP, c'est comme si vous parliez à quelqu'un qu'il soit éveillé ou endormi. S'il est éveillé tant mieux, il vous écoute, il **reçoit votre message**. Par contre s'il dort, tant pis, vous aurez beau parler, il ne recevra rien. Voilà pourquoi UDP est un protocole dit **orienté transaction**. En d'autres termes, il se contente d'envoyer le message et d'espérer que tout se passe bien.

IV.3.1.3. La structure d'un datagramme UDP

UDP a été créé par David Reed en 1980 et est spécifié par la RFC 768 . Comme tout protocole, il a un en-tête qui lui est propre. Dans le chapitre précédent, nous vous avons schématisé quelques champs d'un segment de protocole de transport. Ces champs sont en fait communs à UDP et TCP. Maintenant, nous allons voir de quoi est constitué un datagramme UDP entier. Pour rappel, voici à quoi ressemble une unité de protocole de transport :

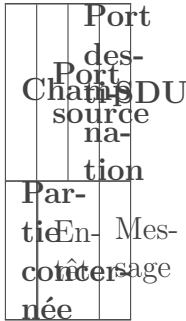


TABLE IV.3.2. – Structure basique d'un protocole de niveau 4

Les champs présents dans le tableau résultent de la fonction de multiplexage et démultiplexage comme nous l'avons vu. Mais UDP ajoute deux champs de plus. Voici la structure d'un datagramme UDP :

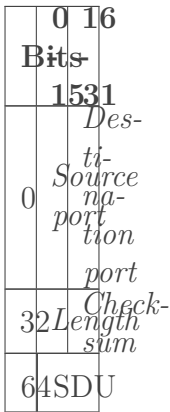


TABLE IV.3.4. – Datagramme UDP

Nous n'allons pas revenir sur les champs « source port » et « destination port », nous y avons déjà passé un bout de temps. Cependant, les deux autres champs « length » (longueur) et « checksum » (somme de contrôle) vous sont inconnus. Chacun des champs de l'en-tête UDP vaut 2 octets, soit 16 bits chacun. 🍊

- **Length** : ce champ spécifie la longueur de tout le datagramme. Un datagramme est le nom que l'on donne à un PDU envoyé par un protocole non fiable, comme nous l'avons

dit dans les chapitres précédents. Étant donné qu'UDP n'est pas fiable, on appelle donc les données à ce niveau « datagramme » (*datagram* en anglais). Ainsi, le champ « *length* » contiendra la longueur du datagramme. Puisque nous savons déjà que chaque champ de l'en-tête vaut 2 octets, il est donc logique que la valeur minimum de ce champ est de 8 octets (2 octets x 4 champs) soit 64 *bits* (16 *bits* x 4 champs).

- **Checksum** : ah, vous vous souvenez de l'expression **somme de contrôle** ? La valeur présente dans ce champ indiquera si une erreur s'est produite durant la transmission du datagramme. Nous verrons comment cela fonctionne dans un prochain chapitre.

##### IV.3.1.4. L'essentiel du protocole UDP

Pour commencer, jetons un coup d'œil aux caractéristiques du protocole UDP :

- **Mode non connecté** : nous avons dit que UDP n'utilisait pas un mode orienté connexion, donc il n'implémente pas le principe de connexion par poignée de main.
- **État de connexion** : l'état de connexion est un sujet un peu complexe pour l'instant, et il est propre à TCP. Alors nous n'allons pas vous faire souffrir pour rien. Retenez juste qu'UDP ne maintient pas des informations sur l'état de la connexion.
- **La longueur de l'en-tête** : comme vous avez eu l'occasion de le voir par vous-même, l'en-tête d'un datagramme UDP est relativement simple. Il ne vaut que 8 octets. Vous allez voir que l'en-tête de TCP est plus complexe que celui d'UDP.
- **Taux de transmission** : avec UDP, la vitesse (ou le taux) de transmission est intimement liée aux caractéristiques de l'environnement du système d'exploitation. C'est-à-dire que cette vitesse dépend des ressources système (fréquence du processeur, etc.) d'une part, et de la bande passante d'autre part. Puisque UDP n'a pas de mécanisme de contrôle de congestion (on en a un peu parlé précédemment), le taux de transmission dépend de la vitesse à laquelle l'application utilisée génère ces datagrammes.

Ces caractéristiques révèlent en quelque sorte les avantages d'UDP. Puisqu'il n'est pas orienté connexion, il ne crée donc pas des délais d'attente pour établir une connexion avant transmission. Grossièrement, imaginez qu'envoyer un message « salut » à un hôte prend 3 secondes. Avec TCP, ça prendra 3 secondes + x secondes, x étant le temps mis pour établir une connexion avant la transmission. C'est un exemple très simplifié. 🍊 Étant donné qu'il ne gère ni l'état de la connexion, ni le contrôle de congestion, il est donc un peu plus rapide que TCP.

Utilisez UDP lorsque vous n'avez pas besoin de garantir que les données envoyées arriveront à leurs destinataires. Par exemple, dans le cadre de la VoIP (la téléphonie via Internet), plusieurs applications utilisent UDP, comme Skype. S'il faut renvoyer les messages vocaux à chaque fois jusqu'à ce qu'ils arrivent, cela va significativement engorger le réseau, voilà pourquoi les conversations téléphoniques sur Internet se font via UDP. Le service DNS (qui permet de traduire les adresses IP en noms de domaine) utilise également UDP.

Nous allons comparer UDP et TCP plus tard, et c'est là que nous allons voir quand il faut utiliser TCP et quand il faut privilégier UDP.

UDP n'étant pas vraiment un protocole très fondamental de la pile TCP-IP, nous n'allons pas nous attarder sur son compte. En revanche, nous allons vraiment nous attarder sur TCP qui a le mérite de se retrouver dans le nom même du modèle de communication sur lequel se base Internet, à savoir **TCP-IP**. 🍊

### IV.3.2. Le protocole pessimiste: TCP

Nous espérons que vous avez encore en mémoire les exemples précédents (transmission des lettres entre cousins des deux maisons, aller à une fête en groupe en empruntant des taxis...). Dans ces deux exemples, nous avons illustré les principes fondamentaux qui sont aussi les divergences principales entre UDP et TCP.

#### IV.3.2.1. TCP dans les grandes lignes

TCP signifie *Transmission Control Protocol*, soit protocole de contrôle de transmission. C'est un protocole de la couche transport défini par les RFC 793, 1122, 1323 et aussi 2581.

Tout comme UDP, le but de TCP est d'assurer la transmission des SDU. Tout comme UDP, TCP nous offre des services de multiplexage et démultiplexage. La différence se situe fondamentalement dans ce que nous avons appelé « principe de transmission fiable ».

Dans notre exemple de transmission des lettres de début de partie, Pierre et Jean représentent un protocole de transmission. Nous avons dit que ces derniers pouvaient offrir un service fiable en garantissant à leurs frères que les lettres arriveraient à la Poste.

Dans notre exemple impliquant les files de taxis, nous avons dit qu'il était impossible de garantir que chaque groupe arriverait à la fête en raison des imprévus.

UDP, comme nous l'avons vu, est un protocole orienté transaction, par conséquent, il ne peut garantir que les données transmises arriveront à destination. TCP, en revanche, est un protocole orienté connexion. Par le principe de la poignée de main (*handshake*), il établit une connexion au préalable entre les processus avant de débiter la transmission.

Le nom de cette section qualifie TCP comme étant un protocole pessimiste. Vous vous demandez certainement pourquoi (faites semblant, au moins 🤔). TCP « n'espère pas » comme UDP que les données arriveront à destination. Au contraire, en protocole responsable, il s'assure que les données transmises arriveront à bon port. Cette garantie est possible grâce à ses fonctionnalités : détection d'erreur, contrôle de séquences, retransmissions, et la plus importante : accusé de réception.



TCP offre des services de détection d'erreurs dans les transmissions, mais n'a aucune propriété de « réparation » (*data recovery* en anglais) de ces données dont l'intégrité est affectée.

#### IV.3.2.2. L'état de connexion avec TCP

Nous avons dit que TCP était un protocole orienté connexion, mais qu'est-ce que cela veut dire concrètement ?

Pour qu'il y ait échange ou transmission entre deux applications utilisant TCP comme protocole de transport, il faut établir une connexion TCP. La poignée de main (*handshake*), le genre de connexion que TCP établit, est aussi appelé « *three-way handshake* ». Cela veut dire que la connexion se fait en trois étapes. Vous vous souvenez des concepts *full-duplex* et *half-duplex* ? Ils s'appliquent aussi ici. En effet, TCP est un protocole *full-duplex*, ce qui implique qu'une communication bidirectionnelle et simultanée est possible dans une communication TCP.



En guise de rappel, un système de communication *half-duplex* veut dire que la communication se fait dans un seul sens à la fois. L'hôte A transmet à l'hôte B ou vice versa, mais en aucun cas les deux peuvent transmettre simultanément. Cet avantage ne nous est offert que dans un système de transmission *full-duplex*, et ça tombe bien : TCP est un protocole *full-duplex*.

Voici comment s'établit une connexion suivant le principe de « *three-way handshake* ». Supposons que 2 processus de deux hôtes différents veulent s'échanger des informations. Il faut, bien entendu, qu'un hôte initialise la connexion (demande à l'autre hôte la permission de communiquer). Comme vous le savez, l'hôte qui initialise la connexion est le client, et celui qui accepte est le serveur. 🍊

Le client et le serveur s'échangent donc des unités de données appelées des segments TCP qui sont composés bien entendu d'un en-tête et d'un espace de données (*data area* en anglais). Pour nous assurer que vous n'aurez pas de difficulté à assimiler le *three-way handshake*, nous allons y aller petit à petit en détaillant chaque étape. Le long de notre analyse, nous allons considérer 6 paramètres extrêmement importants pour saisir la procédure d'établissement d'une connexion TCP entre un client et un serveur. Ces paramètres sont :

- État du serveur ;
- État du client ;
- Segment transmis par le client ;
- Segment transmis par le serveur ;
- État de transition du client ;
- État de transition du serveur.

Les noms des paramètres sont assez évocateurs. L'état du client/serveur définit dans quel état se trouve le client/serveur dans l'établissement de la connexion. Le segment transmis par le client/serveur fait référence au segment TCP constitué d'un en-tête et d'une donnée, que les deux s'échangent le long de la procédure. Finalement, l'état de transition du client/serveur définit simplement un état... entre deux états. 🍊 Un exemple : quand le serveur passe de l'état fermé (*closed*) à l'état ouvert, l'état de transition ici est l'état d'écoute (*listen*). En effet, pour pouvoir ouvrir ou initialiser une connexion, il faut bien que le serveur **écoute** le port auquel il est attaché. On dit que cet état d'écoute est un état de transition. 🍊

Vous êtes prêts ? C'est parti ! 🍊

#### IV.3.2.3. Étape 1

Pour faire une demande d'initialisation de connexion, le client envoie un segment TCP qui ne contient **que** l'en-tête. Il est important de noter que le premier segment TCP envoyé dans le *three-way handshake* n'a **aucun payload** (aucune donnée utile), mais **seulement un en-tête**.



Et il y a quoi dans cet en-tête alors ?

Nous y arrivons ! L'en-tête contient un *flag* (drapeau) de synchronisation : SYN, diminutif de *SYNchronize*, synchroniser. L'en-tête contient aussi le numéro de port TCP du serveur, comme pour UDP.

Après l'envoi de ce premier segment TCP, le client se met dans un état **SYN\_SENT**. Pour ceux qui n'ont pas séché leurs cours d'anglais ( 🍊 ), vous comprenez tout de suite que SYN\_SENT

#### IV. Dans les basses couches du modèle OSI

signifie que le segment TCP ayant le *flag* SYN a été envoyé.

Le serveur, de son côté, est en état **Listen**. Il écoute sur le numéro de port, attendant une demande de connexion. Dès qu'il reçoit le segment TCP envoyé par le client (celui qui n'a qu'un en-tête et un *flag* SYN), il envoie un segment TCP vers le client pour confirmer sa réception. Ce segment n'est constitué que d'un en-tête portant deux *flags* : **SYN** (*synchronize*) et **ACK** (*acknowledge*, accusé de réception).

Finalement, le serveur change d'état et passe de **Listen** à **SYN\_RCVD**, RCVD étant le diminutif de *received* (reçu en anglais). Cet état confirme que le segment TCP portant le *flag* SYN a été reçu.

i

SYN ou ACK sont deux *flags* importants dans l'en-tête d'un segment TCP. Dans un en-tête TCP, toute une zone est dédiée à ces *flags*. À chacun d'entre eux correspond un *bit* spécifique. Ainsi, quand un *flag* est spécifié (on dit aussi que le drapeau est levé), cela se traduit dans l'en-tête par l'allumage du *bit* correspondant, qui vaut alors 1. Si le drapeau n'est pas levé, le *bit* vaut 0. Nous allons voir la structure détaillée un peu plus loin, pas de panique ! 🍊

#### IV.3.2.4. Étapes 2 et 3

Le client reçoit le segment TCP et analyse l'en-tête. Il remarque qu'il contient les *flags* SYN et ACK, qui sont pour lui des indicateurs que le serveur a été gentil accepte d'initialiser une connexion avec lui. La connexion passe donc de l'état **Closed** (fermée) à **Established** (établie). Il conclut la procédure en envoyant un segment TCP avec le *flag* ACK au serveur en guise de remerciement. 🍊 Bon, en vrai, ce dernier ACK envoyé est la conclusion de la procédure et va également mettre le serveur en état de connexion établie (*established state*).

Voilà ! Maintenant que les deux hôtes (client et serveur) ont établi une connexion, cette dernière peut être utilisée pour transmettre des informations !

i

Dans notre scénario, c'est le client qui est responsable d'avoir fait une demande de connexion. On dit que le client effectue une **connexion active** (*active open*) alors que le serveur n'a fait qu'accepter l'invitation, ce dernier effectue une **connexion passive** (*passive open*). 🍊

En résumé, voici une image pour vous permettre de mieux visualiser la procédure.

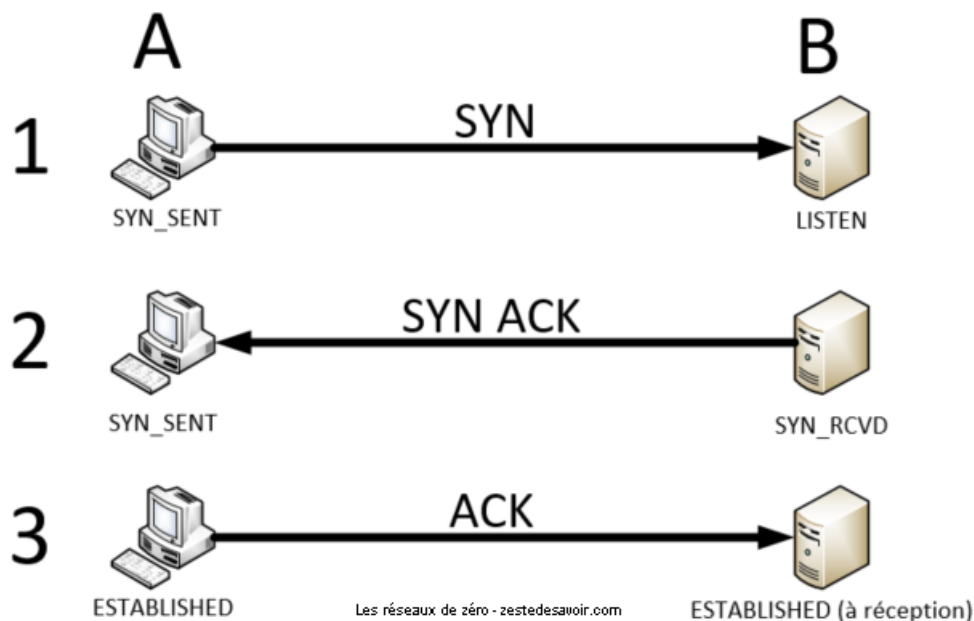


FIGURE IV.3.2. – Schéma illustrant l'établissement d'une connexion TCP

Voilà pour l'établissement, mais maintenant, il faut transmettre des données ! À quoi cela sert d'être connecté, sinon ? 🍊

#### IV.3.2.5. Séquence émotion et acquittement

Comment TCP s'assure que les données sont bien arrivées ? Grâce à des numéros de séquence ! Lorsqu'une connexion TCP est initialisée, un nombre est généré : l' **ISN** (cela peut être un zéro ou un nombre aléatoire suivant les systèmes). Quand des données sont envoyées, ce numéro de séquence est augmenté d'autant d'octets de données qui sont envoyés.

Voici un exemple :

2	1.395759	192.168.1.56	192.168.1.62	TCP	66	50591 > 32123 [SYN, Seq=0 win=8192 Len=0 MSS=1460 WS=0
3	1.399660	192.168.1.62	192.168.1.56	TCP	66	32123 > 50591 [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0
4	1.399714	192.168.1.56	192.168.1.62	TCP	54	50591 > 32123 [ACK] Seq=1 Ack=1 win=17520 Len=0
12	4.284998	192.168.1.56	192.168.1.62	TCP	60	50591 > 32123 [PSH, ACK] Seq=1 Ack=1 win=17520 Len=6
13	4.291709	192.168.1.62	192.168.1.56	TCP	54	32123 > 50591 [ACK] Seq=1 Ack=7 win=5840 Len=0
18	7.535097	192.168.1.56	192.168.1.62	TCP	61	50591 > 32123 [PSH, ACK] Seq=7 Ack=1 win=17520 Len=7
19	7.540364	192.168.1.62	192.168.1.56	TCP	54	32123 > 50591 [ACK] Seq=1 Ack=14 win=5840 Len=0
22	9.512235	192.168.1.56	192.168.1.62	TCP	54	50591 > 32123 [RST, ACK] Seq=14 Ack=1 win=0 Len=0

FIGURE IV.3.3. – Capture d'échange de données avec TCP

Sur cette capture, nous pouvons voir que l'hôte 192.168.1.56 a établi une connexion TCP avec 192.168.1.62. Les trois premières trames représentent le *3-way handshake*. La quatrième trame, en bleu, est un envoi de données au travers de cette connexion. Regardez les valeurs de *Seq* et *Len*. Nous partons d'un numéro de séquence qui est de 1 et nous envoyons 6 octets de données. Le serveur nous répond avec un **ACK** (accusé de réception), ce qui signifie qu'il a bien reçu quelque chose. Il précise la valeur 7 dans le champ « *Ack* », car il a additionné le numéro de séquence du segment reçu (1) avec la longueur des données du segment en question (6), ce qui fait bien 7. Le client sait donc que son segment a bien été reçu.

Que se serait-il passé si le segment n'avait pas été reçu ? Pour une raison quelconque, il aurait pu être perdu en cours de route. Dans ce cas, si le client ne reçoit pas d'accusé de réception, il renvoie automatiquement le même segment au bout d'un temps défini appelé **RTO**. Ce temps est calculé dynamiquement. Si vous voulez vous amuser à comprendre ce mécanisme, vous pouvez lire la [RFC 2988](#) .

?

Lors de l'établissement en trois phases, aucune donnée n'est transmise, et pourtant, le serveur répond avec un *Ack* de 1 alors que le numéro de séquence du segment envoyé est 0 !

Si vous l'avez remarqué, chapeau ! Effectivement, le numéro de séquence des deux machines est augmenté de 1 lors du *3-way handshake*.

Le principe des numéros de séquence et d'acquiescement est fondamental pour comprendre TCP. Pourtant, ce protocole a plus d'un tour dans son sac et permet d'effectuer tout un tas de contrôles pour optimiser le traitement des communications. Avant d'aller plus loin, voyons comment est structuré un segment TCP pour visualiser toutes les possibilités qu'il offre.

#### IV.3.2.6. Structure d'un segment

Visualisons les différents champs d'un segment TCP :

Off-set								1								2								3											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
0	0	Port source														Port destination																			
4	32	Numéro de séquence																																	
8	64	Numéro d'acquiescement																																	
12	96	Taille en-tête				Ré-servé				E	C	E	U	A	P	R	S	F	Fe-nêtre																
16	128	Somme de contrôle														Poin-teur de données ur-gentes																			

2016	Op- tion(s) (fa- cul- ta- tif)
se- lon op- tions	Don- nées

TABLE IV.3.6. – Tableau représentant les champs d'un segment TCP

La plupart des informations transmises par TCP comme la fenêtre ou la somme de contrôle seront étudiés dans les prochains chapitres. Pour conclure cette sous-partie, nous allons étudier la fermeture d'une connexion TCP.

Maintenant que nous savons comment s'ouvre une connexion TCP et que nous avons évoqué en surface l'acquiescement des données transmises, il est temps de clore la connexion. Il y a deux façons de faire : la normale... et la brutale. 🐱 Commençons par la méthode douce.

#### IV.3.2.7. Clôture normale

Rappelez-vous ce que deux personnes font quand elles rentrent en contact : une poignée de main. Et que font-elles lorsqu'elles se séparent ? Eh bien, une autre poignée de main. Avec TCP, c'est pareil, c'est juste un peu plus long. Alors que le *handshake* d'initialisation se fait en 3 temps, celui de clôture se fait en **4 temps**.

Considérons deux hôtes A et B qui ont initié une connexion TCP. Peu importe qui est client ou serveur, cela ne change rien. Quand ils n'ont plus rien à se dire, l'un va prendre l'initiative de clore la communication. Dans notre exemple, disons qu'il s'agit de A. Pour cela, il va envoyer un segment vide, comme pour l'ouverture de connexion, mais en levant le *flag* FIN. L'état de la connexion chez l'hôte A passe alors à **FIN\_WAIT\_1**.

L'hôte B va alors répondre avec un segment vide et le *flag* ACK levé. C'est comme la 3ème étape de l'ouverture. L'état de la connexion chez B passe alors à **CLOSE\_WAIT**.

À ce moment-là, A a dit à B qu'il souhaitait clore la connexion, et B a confirmé avoir reçu cette demande. **Il n'a pas encore accepté la clôture**. Pour ce faire, il doit à son tour envoyer un message de fin, que son interlocuteur doit acquiescer.

La 3ème étape est donc l'envoi par l'hôte B d'un segment vide avec le *flag* FIN levé. L'état de la connexion chez B devient alors **LAST\_ACK**. De manière assez explicite, cela signifie qu'il n'attend plus qu'un dernier ACK.

C'est la 4ème et dernière étape : l'hôte A confirme la réception du dernier segment avec un ACK. Lorsqu'il est reçu par B, l'état de la connexion chez B est **CLOSED**. Pour lui, la connexion est proprement fermée. Au moment de l'envoi, l'état de la connexion chez A passe par la phase transitoire **TIME\_WAIT**. Il s'agit d'une période de temps durant laquelle l'hôte patiente pour être sûr que B a bien reçu le segment. Comme il n'y a plus aucun échange après, A considère que B a reçu l'ACK au bout d'une durée égale à 2 MSL. Le MSL (*Maximum Segment Lifetime*) est un paramètre du système qui vaut généralement 30 secondes ou 1 minute.

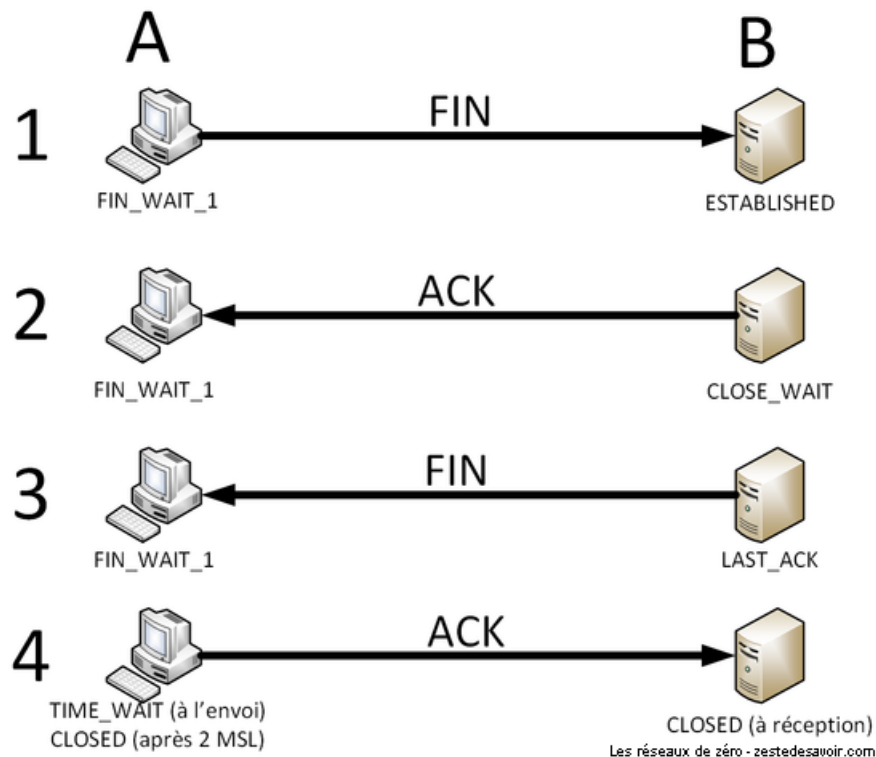


FIGURE IV.3.4. – Processus de clôture normale d'une connexion TCP

Depuis un système Windows ou Linux, on peut visualiser l'état des différentes connexions de notre système avec la commande `netstat`. Voici un extrait de ce qu'elle peut nous retourner (les adresses IP ont été modifiées) :

1	TCP	127.0.0.1:63146	DESKTOP-XXXXXXX:63145	ESTABLISHED
2	TCP	192.168.1.14:62442	ec2-94-107-164-227:https	ESTABLISHED
3	TCP	192.168.1.14:62953	192.125.106.67:27021	ESTABLISHED
4	TCP	192.168.1.14:63217	48.97.221.206:https	ESTABLISHED
5	TCP	192.168.1.14:63266	65:imap	CLOSE_WAIT
6	TCP	192.168.1.14:63275	65:imap	CLOSE_WAIT
7	TCP	192.168.1.14:63282	14.215.39.17:https	ESTABLISHED
8	TCP	192.168.1.14:63283	41.70.165.13:https	TIME_WAIT
9	TCP	192.168.1.14:63284	140.11.113.150:https	ESTABLISHED
10	TCP	192.168.1.14:63285	123.17.31.28:https	ESTABLISHED

Ça, c'est si tout se passe bien (et ça se passe généralement bien). Mais que se passe-t-il en cas de pépin ?

#### IV.3.2.8. Clôture brutale

Il existe un *flag* TCP qui fait partie des plus courants : **RST** (*reset*). RST signifie à peu près « va te faire voir ». 🍊 Ce drapeau apparaît lorsqu'une communication non autorisée arrive sur un port. Typiquement, si vous essayez d'ouvrir une connexion TCP avec un hôte sur un port

#### IV. Dans les basses couches du modèle OSI



quelconque, sitôt votre premier SYN envoyé, vous allez recevoir un segment vide avec seulement le *flag* RST levé (et un ACK pour préciser à quoi on répond). Pourquoi ? Parce que le système que vous contactez va recevoir votre demande de connexion, va regarder le port destination, et si aucun programme n'écoute sur ce port, il n'a aucune raison d'accepter et vous le fait savoir sans détour.

i

Dans ce même cas de figure, il est aussi possible que vous ne receviez rien du tout. Cela arrive lorsque l'hôte distant est protégé par un pare-feu. Ce dernier peut être configuré pour ignorer les demandes de connexion sur des ports inconnus. Cela permet d'éviter d'ennuyer les serveurs avec des requêtes inutiles.

Il peut aussi arriver qu'une connexion déjà établie soit brutalement interrompue par un *flag* RST. Cela peut se produire en cas de mauvais comportement d'un hôte. Un programme peut être conçu pour mettre fin à une connexion s'il détecte que son interlocuteur envoie des données inattendues : requêtes mal formées, brute force de mot de passe, ... Dans ce genre de cas, mieux vaut ne pas prendre de pincettes.

i

Pour aller plus loin, nous vous conseillons l'outil [socklab](#) . Il s'agit d'un programme français très utilisé dans les universités pour l'expérimentation sur TCP et UDP. Vous trouverez une documentation en suivant ce lien [ce lien](#) .

Nous venons d'évoquer les principes de base de TCP. Oui, de base. Les notions plus complexes viendront plus tard. 🐱 Avant cela, reposons-nous un peu la tête en regardant l'affrontement entre TCP et UDP.

### IV.3.3. TCP / UDP : le clash !

Eh bien, en réalité, il n'y a pas de clash puisque ces deux protocoles ne sont pas concurrents, mais complémentaires. C'est comme imaginer un clash entre un rhinocéros et un aigle. Et non, dans Pokémon ça ne compte pas. 🍊

Les usages sont fondamentalement différents. Le choix de l'un ou l'autre peut venir de plusieurs facteurs.

#### IV.3.3.1. Nécessité fonctionnelle

En premier lieu, une application peut tout bonnement **ne pas fonctionner** si son protocole n'est pas transporté de manière appropriée. Prenons pour exemple la visioconférence. Pour que l'image et la voix de votre interlocuteur vous parviennent de manière fluide, il faut que les données transitent rapidement et sans gigue. La gigue (en anglais *jitter*) est le phénomène de décalage, de délai entre l'émission et la réception d'un flux. Le délai de transmission en soi n'est pas forcément gênant, mais si ce délai se met à s'allonger ou se raccourcir brusquement, la vidéo sera saccadée ou, *a contrario*, passera en accéléré pendant un court temps.

Si nous faisons passer un flux de visioconférence sur UDP, que se passera-t-il ? La vidéo va être découpée en petits morceaux au fur et à mesure, puis envoyée directement sans certitude que chaque pièce soit bien reçue. Pour le récepteur, il est donc possible que les morceaux n'arrivent pas tous dans le même ordre voire que certains n'arrivent pas du tout. Si ce cas ne se

#### IV. Dans les basses couches du modèle OSI

produit pas souvent, ce n'est pas forcément gênant : la perte de 100 millisecondes de voix ou d'image n'empêche généralement pas la compréhension par le destinataire, même si cela arrive toutes les 10 secondes. Dans ce cas, c'est de la qualité du réseau que dépendra la qualité de la communication.

Maintenant, que se passerait-il si le même flux était transmis sur TCP ? La moindre donnée perdue devrait être retransmise. TCP considère qu'un segment est perdu lorsqu'il n'a pas reçu d'accusé de réception au bout d'un temps donné, le RTO, qui est de l'ordre de la seconde. Un problème de taille se pose alors : l'hôte destinataire n'enverra pas d'accusé de réception tant qu'il n'aura pas reçu ce paquet perdu, **même s'il a reçu tous ceux qui suivent**. En l'absence de cet acquittement, l'expéditeur va ralentir le rythme, voire arrêter sa transmission. Catastrophe ! Le flux se retrouve alors interrompu temporairement. Quand il reprendra, le délai entre émission et réception se sera élargi, causant ainsi une gigue. Pour compenser cet écart, l'application doit se débrouiller pour passer un moment en accéléré, couper un morceau, etc. De plus, la transmission sur TCP encombre davantage le réseau : les en-têtes sont plus volumineux que ceux de UDP, et les acquittements occupent aussi le support.

La transmission de visioconférence sur TCP n'a donc presque aucun avantage pour beaucoup d'inconvénients, pouvant aller jusqu'au non fonctionnement de l'application. Dans ce cas de figure, UDP s'impose, malgré quelques faiblesses qui peuvent être plus facilement compensées par l'application.

Le cas inverse se rencontre avec d'autres protocoles. Prenons l'exemple de SMTP, que nous avons étudié précédemment. Imaginons qu'un e-mail un peu volumineux, avec des pièces jointes, soit transmis sur UDP. Le message devra être découpé en plusieurs datagrammes. Que se passerait-il si l'un d'entre eux se perdait ? L'expéditeur ne pourrait pas savoir si son courriel a été correctement reçu. Le serveur de messagerie recevrait un message inconsistant, avec éventuellement une pièce jointe corrompue, un contenu dont la taille ne coïncide pas avec celle déclarée dans l'en-tête du mail, ... On pourrait imaginer que le serveur renvoie un message à l'expéditeur pour lui dire que quelque chose cloche, mais d'une part, **ce n'est pas son rôle**, d'autre part, on ne pourrait même pas s'assurer que le message d'erreur arrive à bon port. En revanche, ces problématiques sont très bien gérées par TCP. Ce dernier s'impose donc comme protocole de transport pour SMTP.

##### IV.3.3.2. Nécessité matérielle

On peut aussi se retrouver contraint d'utiliser TCP ou UDP pour des raisons matérielles. Nous avons évoqué le fonctionnement de BitTorrent précédemment. Ce protocole utilise TCP pour la transmission des données (les fichiers échangés), et UDP pour la transmission d'informations de contrôle. En 2008, la société BitTorrent, qui gère le protocole du même nom, a envisagé la possibilité de transmettre les données sur UDP. À cette période, ce système d'échanges P2P était très utilisé, bien plus que de nos jours (2018). UDP n'offrant pas de mécanismes de contrôle de flux ou de congestion, contrairement à TCP, des craintes sont apparues quant à la saturation de réseaux. [Cet article de NextINpact](#) résume la problématique soulevée à l'époque. Il ne s'agissait pas là d'une question de fonctionnement de l'application, mais de savoir si les infrastructures matérielles étaient capables de supporter le changement. Finalement, c'est toujours TCP qui transmet les fichiers échangés sur BitTorrent.

### IV.3.3.3. Complémentarité

Nous avons dit que TCP et UDP étaient complémentaires. En réalité, ils le sont tellement qu'un même protocole peut utiliser **les deux**.

?

Mais comment est-ce possible ? 🍌

Comprenons-nous bien, il ne s'agit pas d'utiliser les deux en même temps. L'un ou l'autre peut être utilisé en fonction du contexte ou des données. Dans le paragraphe précédent sur BitTorrent, nous avons évoqué le fait que TCP était utilisé pour les **contenus** et UDP pour les **informations**. La différence se fait sur les données à transmettre. Il existe aussi des cas où la différence est contextuelle.

Le protocole DNS (*Domain Name System*) permet de faire le lien entre noms de domaines, comme `zestedesavoir.com`, et adresses IP, comme `2001:4b98:dc0:41:216:3eff:febc:7e10` (un peu d'IPv6 pour changer ! 🍌). Quand un hôte demande à un serveur DNS une adresse IP, l'échange se fait sur UDP. Par contre, lorsque deux serveurs DNS font partie d'un même ensemble (l'un étant le *backup* de l'autre par exemple, ce qu'on appelle une **relation maitre-esclave**), les requêtes DNS entre ces serveurs se font en TCP.

?

Pourquoi ça ne se fait pas en UDP comme pour les requêtes habituelles ?

C'est une question de contexte. Quand votre ordinateur envoie une requête DNS ou reçoit une réponse, le contenu est minimaliste. Le paquet tout entier va faire quelques centaines d'octets tout au plus. Si on l'expédiait sur TCP, le volume d'échanges pour une seule requête serait au minimum doublé : d'abord le *3-way handshake*, le paquet contenant la requête serait alourdi de tout l'en-tête TCP, le serveur devrait répondre d'un acquittement même s'il n'a pas de réponse à fournir, sans compter la clôture propre de la connexion. Le jeu n'en vaut pas la chandelle, on préfère envoyer les requêtes DNS sur UDP quitte à devoir en retransmettre une de temps en temps en cas de perte.

Toutefois, quand deux serveurs travaillent en binôme, il est primordial qu'ils soient synchros sur les réponses qu'ils fournissent aux clients. On ne peut pas se permettre que l'un soit au courant d'une correspondance nom de domaine-adresse IP et pas l'autre : cela serait bancal pour les utilisateurs. On a besoin, dans le cadre de cette relation maitre-esclave, de **fiabilité**. Dans ce contexte, le volume d'échanges peut parfois être important (dans le cas de transfert de zones notamment), et le besoin de transmission fiable est réel, on utilise donc TCP.

Voici un schéma illustrant cela. Notez que lorsqu'une requête DNS d'un client est particulièrement volumineuse, on utilise aussi TCP, bien que ce cas soit rare.

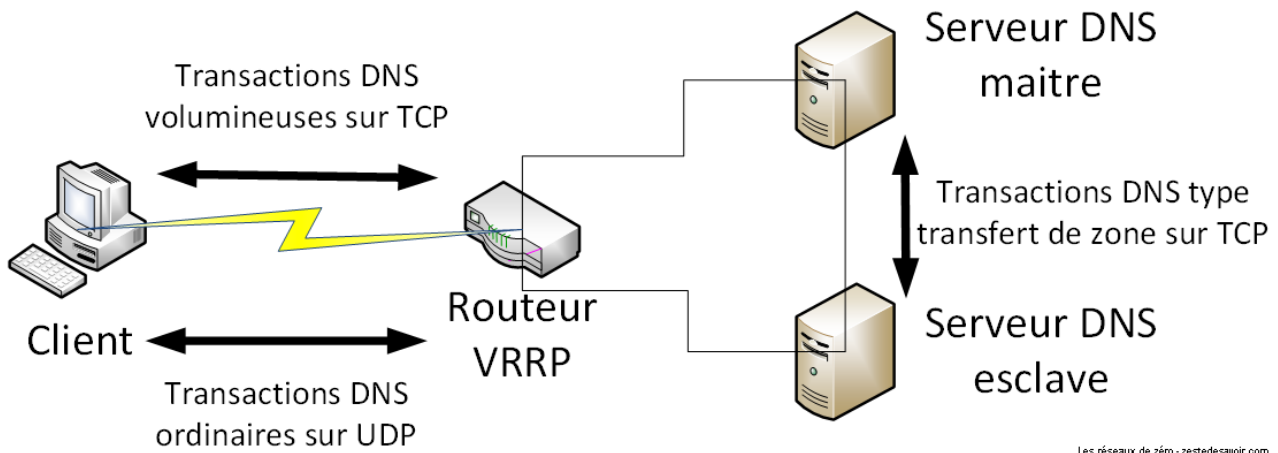


FIGURE IV.3.5. – Protocoles de transport de DNS

i

Dans ce schéma, on peut voir un routeur qui sert d'intermédiaire entre les serveurs DNS et le client. Il fait le relai dans le cadre du VRRP, un protocole qui permet de basculer d'un serveur à l'autre en cas de problème ou pour équilibrer la charge.

## Conclusion

Ce chapitre nous a permis d'étudier les faiblesses du protocole UDP et surtout de comprendre pourquoi TCP est le protocole de transport le plus utilisé. En outre, vous avez saisi que dans certaines circonstances, il est préférable d'utiliser UDP. Le but de la connaissance théorique des protocoles est justement de vous permettre d'être en mesure de choisir celui qui correspond le mieux aux contraintes de votre projet. TCP, comme nous l'avons souligné, est un protocole très riche, c'est pour cela que nous ne pouvons pas évoquer d'un seul coup toutes les possibilités qu'il offre.

## IV.4. Aujourd'hui, contrôle !

### Introduction

Sortez une feuille et un crayon, contrôle surprise ! Comment cela, ce n'est pas crédible ? 🍊  
Nous avons vu que TCP permettait d'assurer le bon déroulement d'une transmission grâce à divers mécanismes. Nous allons détailler certains d'entre eux pour comprendre comment ce protocole fait pour assurer que les données parviennent en toute intégrité et de manière fluide.

#### IV.4.1. Le contrôle de flux

Un des mécanismes proposés par TCP est le contrôle de flux. Il permet de réguler la vitesse de transmission au sein d'une session TCP.

?

Quel intérêt ? Ce qu'on veut, c'est aller le plus vite possible !

Cela peut sembler paradoxal, mais pour aller le plus vite, il ne faut pas se lancer à pleine vitesse. Sur une route, si toutes les automobiles se mettent à rouler à leur vitesse maximale, au moindre obstacle ou virage, cela va provoquer des freinages brusques, des ralentissements, des collisions... Et ce, même si la route est large ! Au final, entre celles qui ne sont jamais arrivées, celles qui sont parties très vite mais se sont retrouvées ralenties derrières d'autres très lentes ou dans les bouchons, et les quelques-unes qui s'en sont sorties, la vitesse moyenne est hasardeuse. Tandis que si tout le monde se met d'accord pour ne pas dépasser une vitesse donnée, disons 100 km/h, et pour que ceux qui ne peuvent pas atteindre cette vitesse restent sagement sur la file de droite, on aura une vitesse moyenne de 100 km/h pour les plus rapides, et une vitesse moyenne qui correspondra à leur vitesse maximale pour les plus lents. Et on évite ainsi bouchons et accidents !

?

Quel rapport avec le réseau ?

Au niveau transport, UDP ne contrôle pas sa vitesse. Quand un datagramme part sur le réseau, il part, point. Si un million de datagrammes veulent partir simultanément et que la machine de l'émetteur le permet, ils partiront. Le problème va se poser à l'arrivée. Que se passe-t-il si tout arrive en même temps chez un destinataire qui ne peut pas tout traiter, par exemple parce qu'il a déjà beaucoup de trafic à gérer ? Les premiers datagrammes pourront être traités, une partie pourra être stockée dans une **mémoire tampon** le temps d'écouler des flux, mais cette mémoire n'est pas illimitée. Une fois remplie, tout le reste est refusé par le destinataire et les données sont tout bonnement perdues. Nous avons vu précédemment qu'UDP n'est pas utilisé pour le transport de gros volumes pour ce genre de raisons.

TCP permet de pallier ce manque. Le **contrôle de flux** a pour but d'adapter la vitesse en

IV. Dans les basses couches du modèle OSI

fonction des performances du destinataire. Pour cela, 3 paramètres d'en-tête interviennent : le **numéro de séquence**, le **numéro d'acquittement** et la **fenêtre**.

Offset								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0 0		Port source														Port destination															
4 32		Numéro de séquence																													
8 64		Numéro d'acquittement																													
12 96		Taille en-tête				Ré-servé				E	C	E	U	A	P	R	S	F	Fe-nêtre												
16 128		Somme de contrôle														Poin-teur de don-nées ur-gentes															
20 160		Op-tion(s) (fa-cul-tatif)																													
se-lon op-tions		Don-nées																													

TABLE IV.4.2. – Rappel de la structure de l'en-tête TCP

IV.4.1.1. Acquitté!

Voyons ce qui se passe du début à la fin d'une transmission. Lorsqu'une session TCP est initiée par un hôte A, un numéro de séquence est généré. Le plus souvent, il est aléatoire. Il est inclus dans l'en-tête. Comme c'est le tout début, il faut réaliser le *3-way handshake* pour ouvrir la

IV. Dans les basses couches du modèle OSI

connexion. L'hôte A envoie donc un segment TCP ne contenant pas de données utiles mais seulement un en-tête, qui contient alors notamment un numéro de séquence (disons 3.000.000) et le *flag* SYN levé.

Offset								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0 0		Port source														Port destination															
4 32		Numéro de séquence : 3.000.000																													
8 64		Numéro d'acquiescement : 0																													
12 96		Taille en-tête				Ré-servé																									
																S Y N															
16 128		Somme de contrôle														Poin-teur de données ur-gentes															

TABLE IV.4.4. – En-tête TCP partiellement rempli, phase 1 du 3-way handshake

Que se passe-t-il chez le destinataire, l'hôte B ? On suppose qu'il accepte la connexion et procède à la 2ème étape de la poignée de main. Il doit donc renvoyer un segment vide avec, dans l'en-tête, un numéro de séquence qui lui est propre. Celui-ci est **totale- ment dissocié de celui de l'hôte A** et peut être généré aléatoirement. Nous décidons arbitrairement que ce sera 15.000.000. Les drapeaux SYN et ACK sont levés, mais qu'est-ce qui est acquitté ? Il s'agit de la demande d'ouverture de connexion. Celle-ci est identifiée par le numéro de séquence du segment, en l'occurrence, 3.000.000. Pour que l'hôte A le sache, on le mentionne dans l'en-tête.



J'ai compris ! On va donc avoir la valeur 3.000.000 dans le champ ACK du segment !

Presque. 🍊 Nous avons évoqué cela dans le chapitre précédent : pour accuser réception d'un segment, on envoie le numéro qui correspond au prochain numéro de séquence attendu. Pour

#### IV. Dans les basses couches du modèle OSI

cela, on ajoute au numéro de séquence reçu le nombre d'octets utiles (de *payload*) reçus, ou bien 1 si le *flag* SYN, RST ou FIN est présent. Cela montre à l'expéditeur quel est le numéro de la prochaine séquence attendue, et par là même, que l'on a reçu toutes les séquences qui précèdent. Dans le cas présent, il n'y a pas de donnée utile mais le *flag* SYN est levé, on ajoute donc seulement 1. La valeur du champ ACK est ainsi 3.000.001.

Offset							1							2							3										
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Port source															Port destination																
Numéro de séquence : <b>15.000.000</b>																															
Numéro d'acquiescement : <b>3.000.001</b>																															
Taille en-tête				Ré-servé						ACK				SYN		Fenêtre															
Somme de contrôle															Pointeur de données urgentes																

TABLE IV.4.6. – En-tête TCP partiellement rempli, phase 2 du 3-way handshake

*i*

Ce système peut sembler abscons. Pourtant, ce simple système de calcul permet d'assurer l'acquittement d'un segment **dans tous les cas**. Les segments SYN ou FIN de poignée de main sans *payload* ? En ajoutant seulement 1, on est sûr de ne pas confondre un SYN ou un FIN avec un autre segment. Un accusé de réception se perd ? Le suivant assurera l'acquittement de tous les octets précédents et compensera la perte. Des données utiles se perdent ? S'il y a un « trou » dans le flux, le destinataire n'acquittera pas plus loin que le début du trou, permettant ainsi à l'expéditeur de savoir quoi renvoyer. Ces cas seront détaillés tout au long de ce chapitre.

IV. Dans les basses couches du modèle OSI

Dans la 3ème phase de la connexion TCP, c'est à A d'envoyer un ACK à B. Comme B a envoyé un ACK de 3.000.001, cela signifie qu'il s'attend à recevoir un segment commençant par ce numéro de séquence. De plus, B a envoyé un numéro de séquence de 15.000.000. Comme il n'y a pas de données utiles mais seulement un SYN, A doit donc s'attendre à ce que la prochaine séquence provenant de B sur cette session porte le numéro 15.000.001. Cela nous donne l'en-tête suivant :

0 0							
---	--	--	--	--	--	--	--

TABLE IV.4.8. – En-tête TCP partiellement rempli, phase 3 du 3-way handshake  
Maintenant, si A transmet sur cette session le message « Bonjour », le numéro de séquence sera toujours 3.000.001. La valeur du ACK ne bouge pas, vu que A s'attend toujours à recevoir la séquence 15.000.001.

Offset								1								2								3							
Offset								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0 0																															

0	0	Port source										Port destination											
4	32	Numéro de séquence : <b>3.000.001</b>																					
8	64	Numéro d'acquittance : <b>15.000.001</b>																					
12	96	Taille en-tête		Ré-servé								A P C S K H				Fe-nêtre							
16	128	Somme de contrôle														Poin-teur de don-nées ur-gentes							
Bon-jour																							

TABLE IV.4.10. – En-tête TCP partiellement rempli, transmission de données utiles  
B reçoit ce segment et l’acquitte. Comme prévu, cette séquence porte le numéro 15.000.001. La valeur de l’ACK est augmentée du nombre d’octets utiles reçus, soit 7. L’ACK vaut donc 3.000.008. Quand A va le recevoir, il comprendra que toutes les données émises avant la séquence 3.000.008 ont bien été reçues. Il peut donc continuer ses envois sans souci.



À quoi sert le PSH dans l’en-tête ?

Quand un segment TCP porte le drapeau PSH (*push*), cela indique que les données doivent être traitées par l’application de destination immédiatement. Il n’y a pas lieu d’attendre une suite pour pouvoir traiter l’information, et donc le segment ne doit pas rester en mémoire tampon.

#### IV.4.1.2. Derrière les fenêtres...

Les éléments que nous venons de voir sont à la base du contrôle de flux, mais ils ne peuvent pas réguler la vitesse d’un flux tous seuls. La **fenêtre** vient les aider.

Nous avons évoqué en début de cette section une **mémoire tampon**. Elle permet de stocker

#### IV. Dans les basses couches du modèle OSI

des paquets reçus sur le réseau en attendant qu'ils soient traités par les applications. Si elle est remplie, les données qui arrivent ne pourront pas être retenues par le système et seront alors perdues. La fenêtre est un champ de l'en-tête TCP qui permet de renseigner l'état de cette mémoire. Dans chaque segment envoyé, le système renseigne la quantité de données qu'il peut garder en stock à un instant donné pour la session. L'interlocuteur ne doit alors pas envoyer davantage d'octets utiles tant qu'au moins une partie n'a pas été acquittée.

Ainsi, si l'hôte B a informé l'hôte A que sa fenêtre est de 900 octets, l'hôte A peut envoyer 3 segments de 300 octets chacun, mais pas davantage. Il doit attendre un acquittement pour être sûr que le tampon de son interlocuteur a commencé à se vider. Lorsque A reçoit l'acquittement du premier segment, il sait que le tampon s'est libéré de 300 octets et peut donc envoyer un segment supplémentaire (on suppose que la fenêtre ne varie pas). Cela permet de maintenir une vitesse d'émission stable, comme on peut le voir sur l'exemple ci-dessous.

Dans l'illustration suivante, on considère que la connexion TCP est déjà établie et que B a informé A que sa fenêtre est de 900. On suppose que cette valeur ne varie pas et que chaque segment contient 300 octets de données utiles.

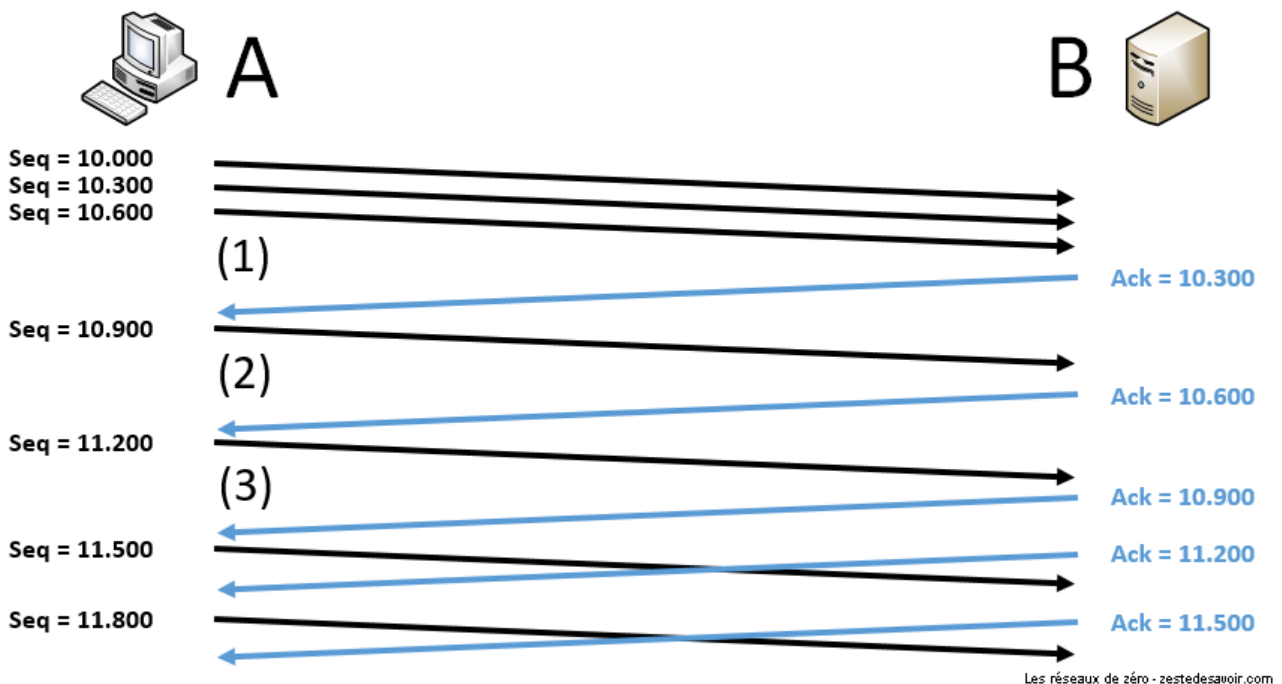


FIGURE IV.4.1. – Représentation d'échanges TCP prenant en compte la fenêtre

Cet échange montre qu'au début, A envoie des segments beaucoup plus vite que B ne peut les traiter. Le pauvre est très sollicité et ne peut pas répondre immédiatement ! A est contraint de ralentir la cadence et d'attendre patiemment de recevoir un accusé de réception avant de continuer. Cela stabilise la vitesse de l'échange.

On parle aussi de **fenêtre glissante** (*sliding window*), car on peut la voir comme une zone qui se déplace le long du flux de données à envoyer. Sur l'illustration ci-dessous, la flèche représente tout ce qui doit être envoyé dans la session TCP. Des traits sont placés tous les 300 octets, le nombre en dessous étant le numéro de séquence correspondant au segment. La fenêtre est représentée en orange et fait toujours 900 octets. On peut voir que sa position évolue entre les points (1), (2) et (3) qui font référence aux points indiqués dans l'illustration précédente.

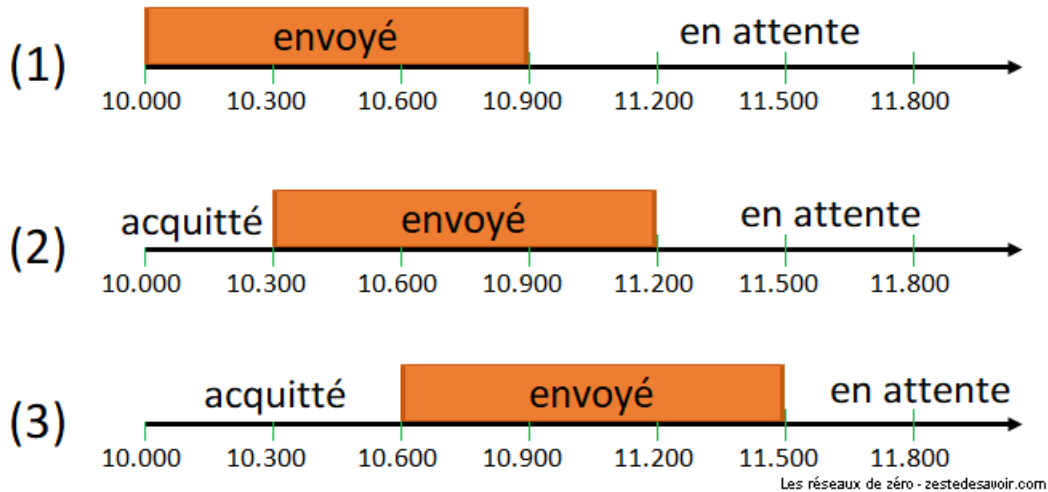


FIGURE IV.4.2. – Position de la fenêtre du point de vue de A

Ce procédé de contrôle de flux permet ainsi de gérer la transmission au niveau d'une session TCP. Il ne prend pas en compte les éléments qui peuvent gêner la communication sur le réseau : c'est le rôle du **contrôle de congestion**.

## IV.4.2. Le contrôle de congestion

Il peut arriver que des données se perdent et n'arrivent pas à destination. Cela se produit notamment lorsque des routeurs, qui font transiter les paquets sur le réseau, sont surchargés. C'est ce qu'on appelle la **congestion**. Ce phénomène se produit au niveau réseau. Pourtant, c'est TCP qui doit se débrouiller pour gérer cette problématique, alors que c'est un protocole de transport. Si le modèle OSI, voire le modèle TCP/IP, était strictement respecté, ce devrait être au protocole IP de s'y coller... Mais non, il n'a pas été conçu comme ça. Qu'à cela ne tienne ! TCP utilise plusieurs mécanismes pour éviter la congestion. Nous n'en verrons que quelques-uns ici afin que vous appreniez les bases du sujet.

### IV.4.2.1. Le premier: Tahoe

C'est en 1988 qu'un premier algorithme a été implémenté dans TCP pour tenter de gérer la congestion. Son nom : Tahoe. Il n'est pas extrêmement performant mais comme il est relativement simple, nous allons l'étudier pour poser quelques bases.

Commençons par définir plusieurs variables. La première sera la **fenêtre de congestion**. Attention, rien à voir avec la fenêtre du contrôle de flux ! La **fenêtre de congestion**, en anglais *congestion window* et souvent abrégée *cwnd*, c'est un paramètre utilisé par le programme qui gère les connexions TCP. Il n'est pas visible, on ne le retrouve pas dans les champs du protocole. Cette variable sera utilisée pour tester différentes vitesses de transmission.

Un autre paramètre est le **seuil de démarrage lent**, plus connu sous le nom de *slow start threshold* et abrégé *ssthresh*. C'est une valeur arbitraire, souvent fixée par défaut à  $2^{16} - 1$ , soit 65535.

L'algorithme Tahoe définit deux phases : *slow start* (démarrage lent) et *congestion avoidance* (évitement de congestion).

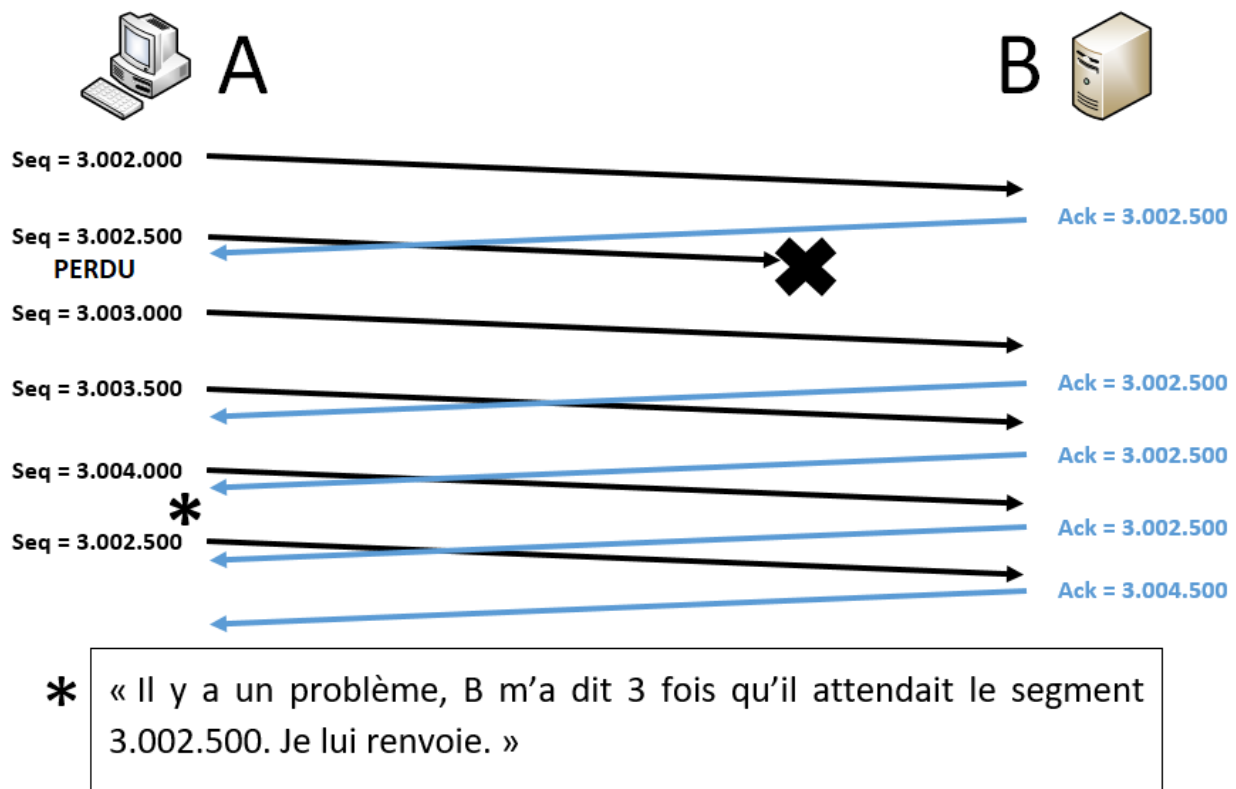
#### IV.4.2.1.1. Slow Start

Pour commencer, on attribue à la fenêtre de congestion (cwnd) la valeur 1. Cela veut dire que l'on n'envoie que 1 segment rempli au maximum, c'est-à-dire 1 fois la taille maximale d'un segment (*Maximum Segment Size*, MSS), et on attend de recevoir un accusé de réception. Quand ce dernier est reçu, on double cwnd. On se permet donc d'envoyer 2 fois le MSS avant d'attendre l'acquittement. Une fois reçu, on double encore cwnd, et ainsi de suite. En cas de perte, le seuil ssthresh est abaissé à la moitié de la dernière cwnd. La fenêtre repart à zéro, et on recommence, jusqu'à ce que la valeur de la fenêtre dépasse le seuil. Ensuite, on quitte le mode *slow start* pour passer en *collision avoidance*.



Comment identifie-t-on une perte ?

Il y a 2 cas de figure. Soit on ne reçoit pas du tout d'accusé de réception et on considère que le paquet est perdu après un temps défini : le RTO (*Retransmission TimeOut*), qui est de l'ordre de la seconde. Soit, au contraire, on reçoit des accusés de réception, mais avec un ACK qui ne bouge pas. Cela signifie que le destinataire continue de recevoir des segments, mais ne peut pas les interpréter car il en manque un morceau, qui est celui désigné par le champ *ACK number*. Dans ce cas, au bout du 3ème accusé de réception identique, TCP comprend qu'il y a un problème avec ce segment et reprend la transmission à ce niveau-là. C'est ce qu'on appelle *fast retransmission*.



Les réseaux de zéro - zestedesavoir.com

FIGURE IV.4.3. – Perte d'un segment de données et fast retransmission

Revenons à notre *Slow Start*. L'illustration suivante représente l'évolution de la fenêtre de congestion au fur et à mesure des transmissions. Pour toutes les images qui suivent, le terme

#### IV. Dans les basses couches du modèle OSI

« transmission » désigne une salve de segments envoyés en même temps. La ligne rouge en pointillés représente le seuil de démarrage lent.

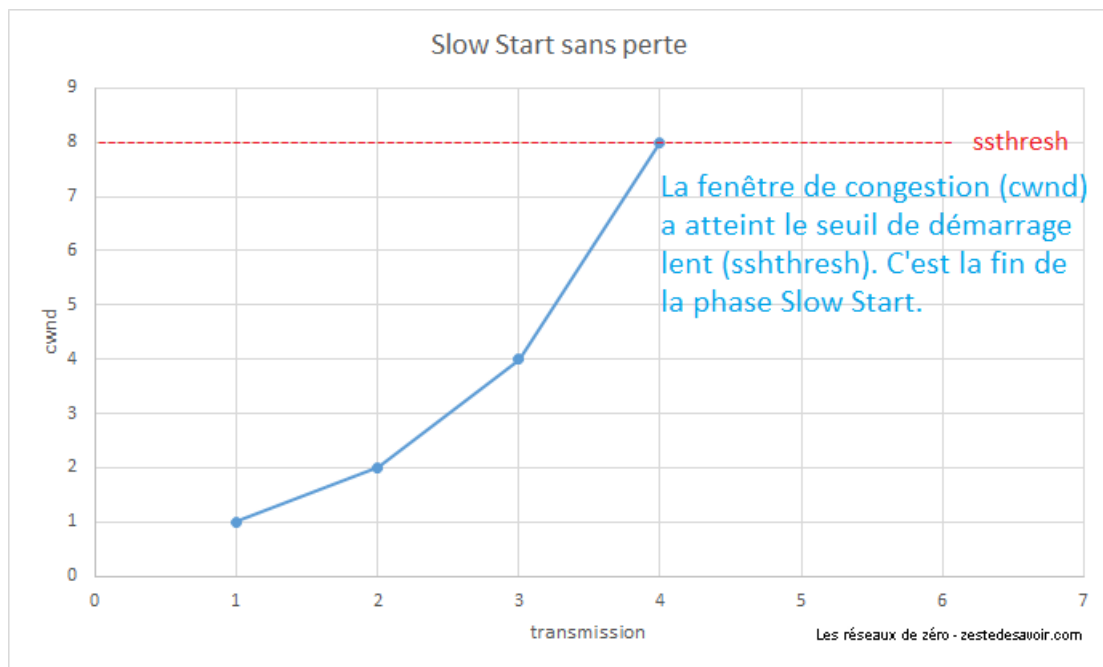


FIGURE IV.4.4. – Évolution de la fenêtre de congestion en Slow Start

Si une perte est constatée avant d'avoir atteint le seuil, celui-ci est divisé par deux et la fenêtre de congestion retombe à 1.

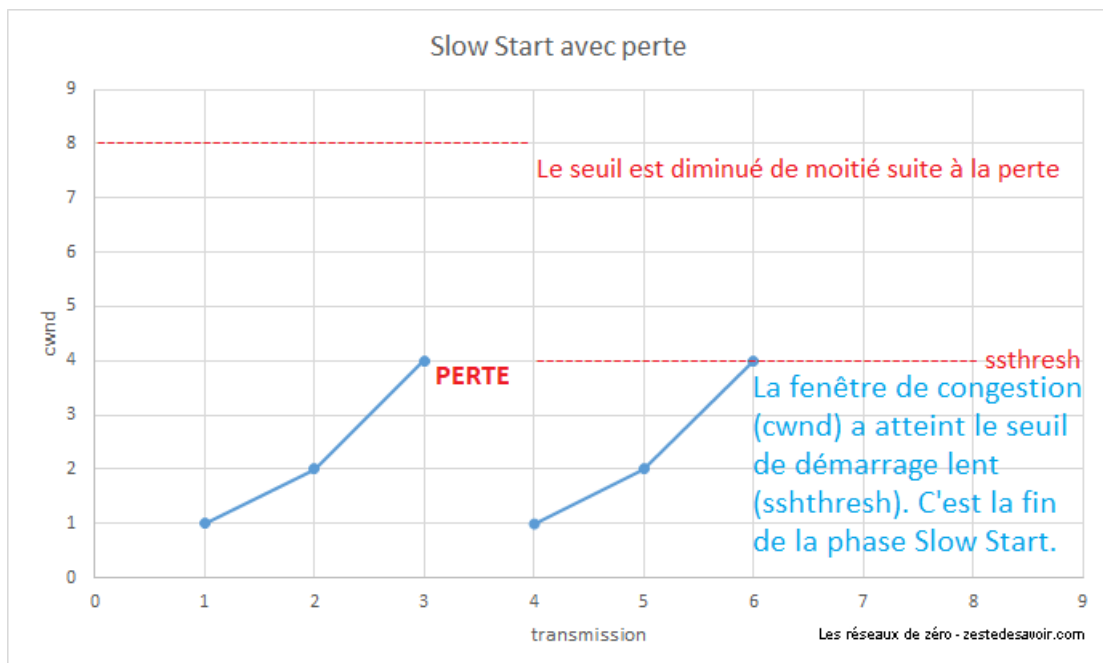


FIGURE IV.4.5. – Évolution de la fenêtre de congestion en Slow Start avec une perte

##### IV.4.2.1.2. Congestion Avoidance

Une fois que la fenêtre a dépassé le seuil, cela signifie que nous ne sommes plus très loin de la vitesse maximale possible. Nous savons *a priori* que nous ne pouvons plus doubler la valeur de la

fenêtre (sauf si le seuil de base a été dépassé). On passe alors en phase *congestion avoidance*. Le paramètre *cwnd* est alors augmenté de 1 à chaque fois, jusqu'à la prochaine perte. Cela permet d'estimer finement une vitesse de transmission maximale. Lorsqu'une perte est constatée, le seuil *ssthresh* est modifié et prend pour valeur la moitié de la dernière fenêtre. La *cwnd* est remise à 1 et on reprend en mode *slow start*.

L'illustration suivante intègre le *slow start*, en trait bleu continu, et la *congestion avoidance*, en pointillés verts.

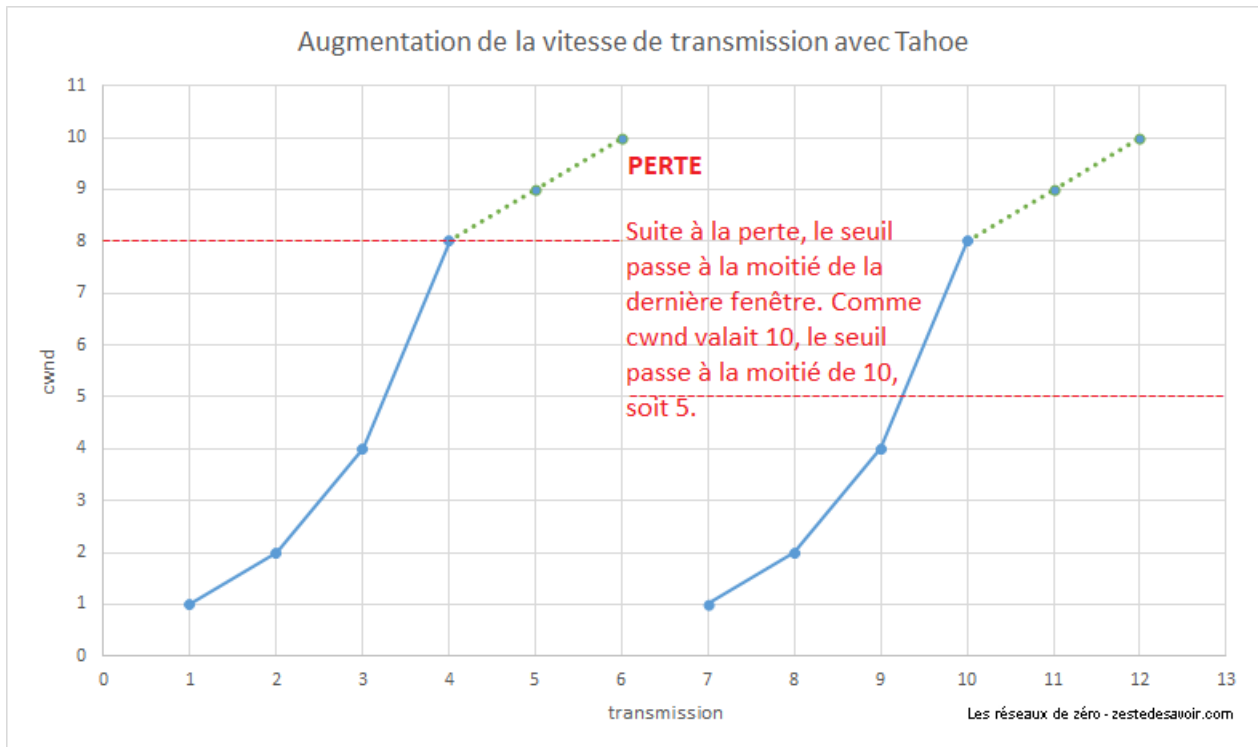


FIGURE IV.4.6. – Slow Start et Congestion Avoidance avec TCP Tahoe

Ce système a le mérite de poser de bonnes bases du contrôle de congestion, mais on visualise bien vite ses limites. Le passage en mode *slow start* à chaque perte provoque une chute brusque du débit, qui met à chaque fois du temps pour revenir à la vitesse d'avant. Il n'est pas possible d'avoir une transmission rapide sans variations régulières. Pour tenter de pallier ces problèmes, un nouvel algorithme a vu le jour 2 ans plus tard : Reno.

#### IV.4.2.2. Reno

Cet algorithme reprend le même mode de fonctionnement que Tahoe, mais il y ajoute le principe de *fast recovery*. Ce mode se déclenche quand un hôte reçoit 3 accusés de réception identiques, indiquant une perte. Quand ce cas se produit, la valeur de la fenêtre est réduite de moitié. Cette nouvelle valeur devient aussi le nouveau seuil *ssthresh*. Une *fast retransmission* est opérée, et l'augmentation linéaire de la fenêtre reprend. Cela permet de réduire les variations de débit en évitant de retomber en *slow start* à la moindre perte. Toutefois, ce mode se réactivera en l'absence totale d'accusé de réception après le RTO.

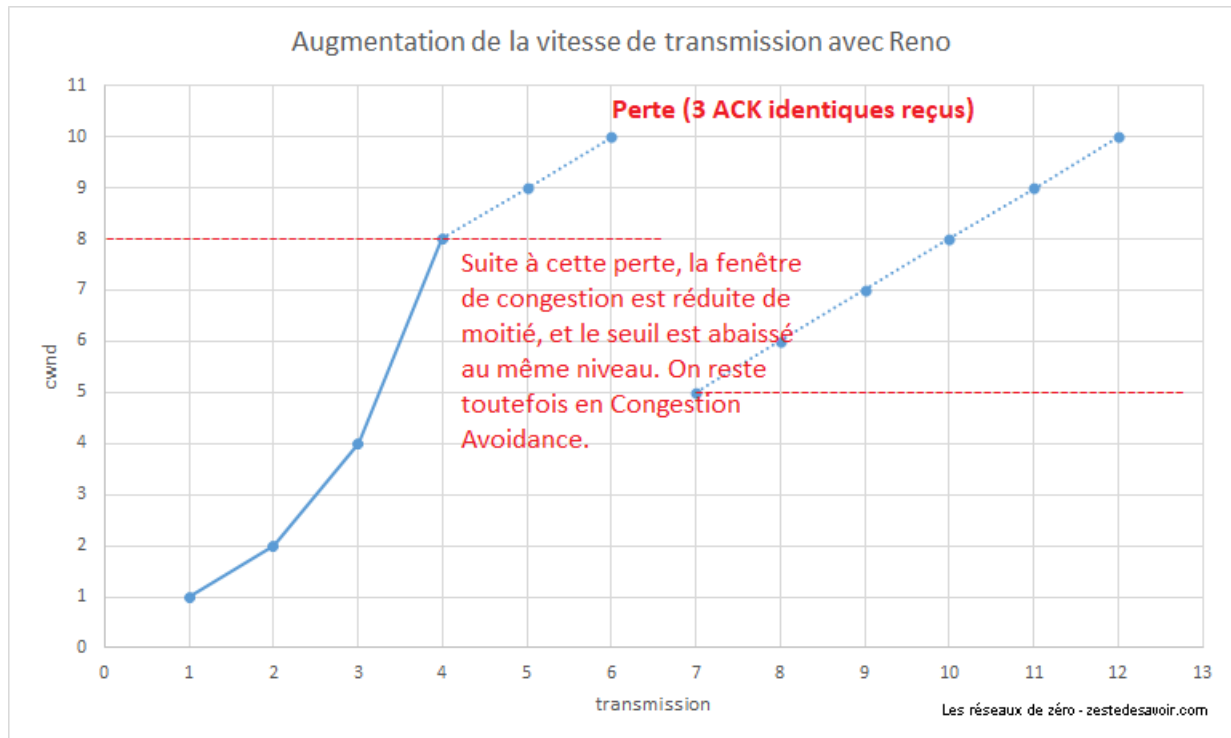


FIGURE IV.4.7. – Évolution de la fenêtre de congestion avec TCP Reno

Reno, ainsi que son évolution, NewReno, ont été beaucoup utilisés dans les années 1990 et au début des années 2000. De nombreux autres ont été inventés. D'autres approches ont été tentées, avec plus ou moins de succès. Les algorithmes Vegas et Westwood+ se basaient sur le temps de transit aller-retour des segments (*round time trip*, RTT) pour ajuster au mieux la fenêtre de congestion. Tous ont été rendus obsolètes par l'augmentation globale des débits.

Depuis 2006, Microsoft utilise [Compound TCP](#) comme algorithme de contrôle de congestion pour ses systèmes Windows, tandis que Linux utilise [CUBIC](#). On ne trouve quasiment pas de documentation à leur sujet en français, si ce n'est leurs pages Wikipédia qui sont assez sommaires. Pour plus de détails, et ce sera en anglais, vous pouvez vous référer à [ce papier très complet de Microsoft](#) pour Compound TCP, ou, pour CUBIC, à [ce document de l'université d'État de Caroline du Nord](#) ou encore à la [RFC 8312](#) (contrairement à la plupart des RFC, celle-ci est une notice d'information et non une spécification).

#### IV.4.2.3. Modification du protocole

En plus des algorithmes, le protocole TCP lui-même a été adapté pour faciliter le contrôle de congestion. Reprenons l'en-tête TCP. Nous pouvons voir au niveau des drapeaux les 3 éléments suivants : ECN, CWR et ECE. Leur fonctionnement est particulièrement astucieux.

La congestion survenant au niveau réseau, quoi de plus à même de signaler la congestion qu'un routeur ? Seulement, un routeur ne doit pas interférer avec des protocoles plus hauts que la couche réseau. Le principe ECN, pour *Explicit Congestion Notification* (notification de congestion explicite), tente de faire passer l'information d'une couche à une autre.

Les routeurs qui supportent l'ECN peuvent, s'ils commencent à saturer, le faire savoir directement dans les paquets IP, notamment ceux qui font circuler les flux TCP. Pour cela, ils vont modifier dans l'en-tête IP un champ particulier. Comme nous n'en sommes pas encore à la couche 3 dans notre cours, nous ne rentrerons pas dans les détails. Cela permet au destinataire d'être informé d'une congestion. Ce dernier, s'il supporte cette fonctionnalité, va en informer l'expéditeur en

#### IV. Dans les basses couches du modèle OSI

levant le drapeau ECE (ECN Echo) dans ses acquittements TCP. L'expéditeur, quand il verra ce *flag*, va réduire de lui-même sa vitesse d'émission, selon les algorithmes dont il dispose. Une fois cela fait, il positionne à 1 le *flag* CWR (*Congestion Window Reduced*, fenêtre de congestion réduite) pour informer le destinataire qu'il a bien reçu l'information et qu'il a pris les mesures appropriées. Le drapeau ECE peut alors être éteint pour la suite des acquittements.

L'illustration suivante suppose que tous les équipements impliqués supportent ECN. Nous partons du principe que la connexion TCP a déjà été initialisée et qu'une congestion est détectée par le routeur (au milieu) lors de la transmission du deuxième segment. On suppose aussi que tout ce matériel peut parler, parce que c'est quand même beaucoup plus simple quand les objets peuvent s'exprimer. 🍊

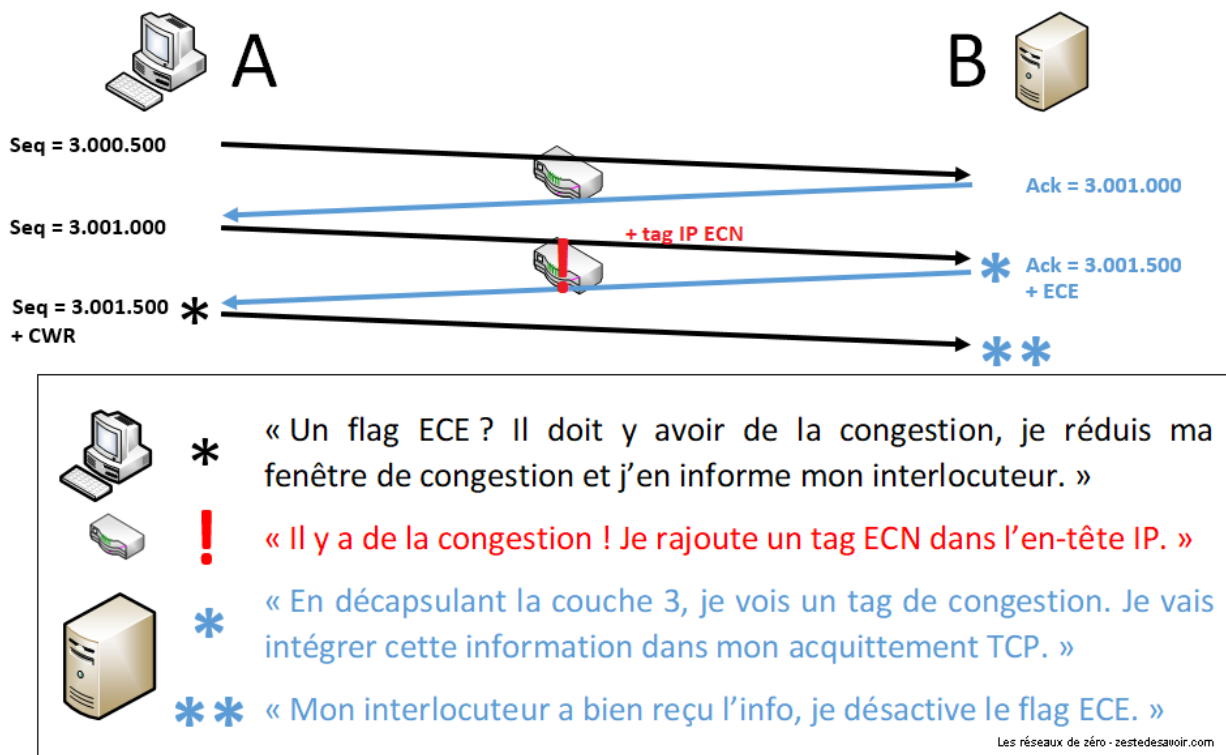


FIGURE IV.4.8. – Principe de fonctionnement d'ECN

i

Lors de l'établissement de la connexion TCP, les hôtes se disent dès le départ s'ils supportent l'ECN. Bien que cette amélioration existe depuis 2001, son adoption à grande échelle s'est faite au milieu des années 2010. Si l'un des deux hôtes ne l'implémente pas ou ne l'a pas activée, l'autre ne cherchera pas à l'utiliser.

Nous avons vu avec le contrôle de congestion que de nombreux mécanismes ont été inventés pour fluidifier au maximum les communications. Cela ne garantit toutefois pas que les données arriveront intègres. Pour cela, on peut utiliser des algorithmes de **somme de contrôle**.

### IV.4.3. Le principe de la somme de contrôle

Cette sous-partie ne sera rien d'autre qu'une petite introduction à la somme de contrôle. Nous ne voulons pas aborder les notions importantes trop rapidement, aussi nous avons préféré consacrer un autre chapitre à la somme de contrôle ainsi qu'à ses différents algorithmes.



C'est quoi, une somme de contrôle ?

La somme de contrôle est un mécanisme utilisé par un protocole afin de détecter des erreurs et de s'assurer que le message reçu est bien intègre, c'est-à-dire qu'il n'a pas subi de modification.

#### IV.4.3.1. Le principe par l'analogie (encore et toujours)

Une fois de plus, nous allons simplement voir en quoi consiste ce principe avec un exemple. Vous vous souvenez de l'exemple avec les cousins et les enveloppes ?

Pour vous rafraîchir la mémoire, nous avons dit que les cousins de « Maison-Est » écrivaient des lettres aux cousins de « Maison-Ouest ». Les lettres étaient dans des enveloppes, et leur grand frère Pierre les collectait afin de les déposer au bureau de poste, qui à son tour, se chargerait de les envoyer dans Ville-Ouest. Si vous ne vous souvenez plus du scénario, vous pouvez relire le début du chapitre précédent.

Nous avons également dit qu'il n'était pas possible de nous assurer que les lettres ne seraient pas modifiées par quelqu'un au niveau de la Poste, n'est-ce pas ? Alors si Junior, un cousin dans Ville-Est, veut s'assurer que Robert, dans Ville-Ouest, reçoive la lettre dans son intégralité, comment peut-il faire ?

Après avoir réfléchi, il décide d'inscrire un motif précis dans la lettre. Il appelle Robert : « Dans ma lettre, à la fin de chaque paragraphe, j'ai mis « it ». Si un paragraphe ne se termine pas par « it », c'est que quelqu'un a modifié cette lettre » (c'est un peu bête et pas fiable, mais l'idée est là 🍊).

La lettre est envoyée et Robert la reçoit. Il s'empresse de vérifier si chaque paragraphe se termine par « it ». Si c'est le cas, alors on peut supposer que la lettre est bien arrivée dans son intégralité. Sinon, il y a eu des erreurs de transmission. 🍊



La somme de contrôle ne peut pas garantir la détection d'erreur à 100% (surtout dans notre exemple bidon). Voilà pourquoi il y a plusieurs algorithmes de calcul de cette somme. Chacun a ses avantages.

En réseau, c'est cela le principe de base de la somme de contrôle. Un protocole (UDP, TCP ou même un autre d'une autre couche) va, par une fonction de computation de somme de contrôle, générer une valeur précise. Cette valeur sera inscrite dans le champ « somme de contrôle (*checksum*) » du protocole utilisé. Lorsque les données arriveront au destinataire, le même protocole va également computer la somme de contrôle avec le même algorithme et comparer la valeur obtenue à celle qui se trouve dans le champ « somme de contrôle » de l'en-tête du paquet. Si les deux valeurs sont les mêmes, alors les données sont intègres, sinon, il y a eu des erreurs de transmission.

Schématiquement, voici comment ça se passe :

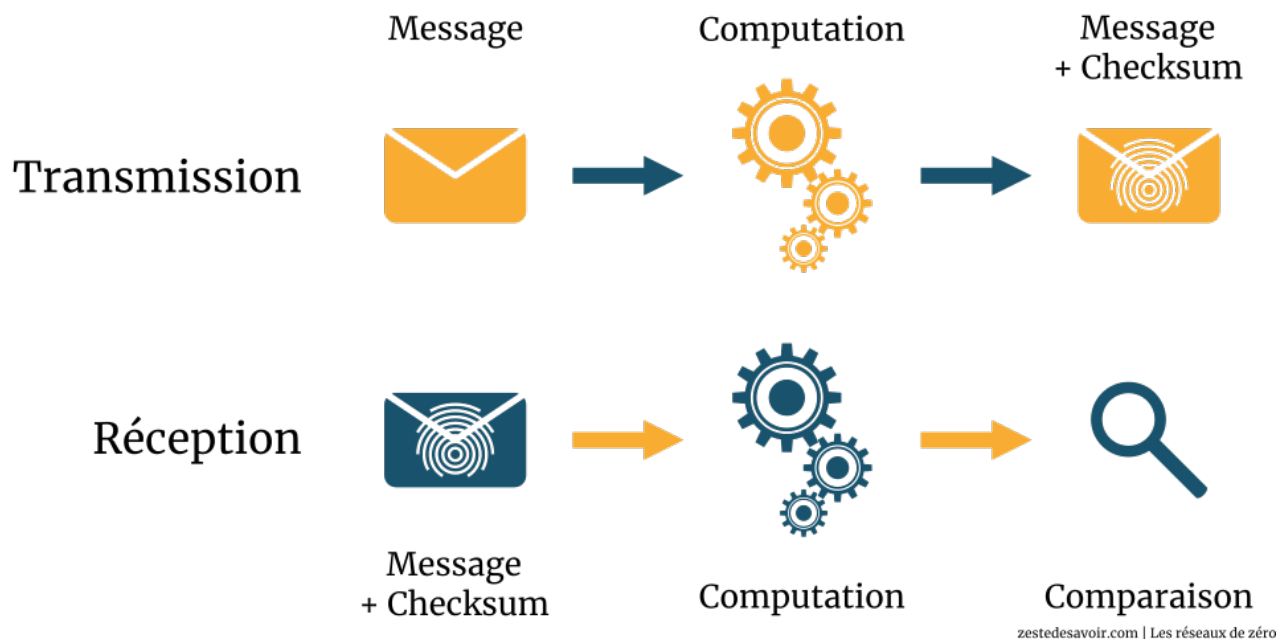


FIGURE IV.4.9. – Utilisation d’une somme de contrôle dans un échange (CC BY)



La somme de contrôle permet uniquement de détecter les erreurs de transmission et non de les corriger.

#### IV.4.3.2. Exemple de computation abstraite

Nous allons essayer de vous expliquer la somme de contrôle par une illustration.



Nous allons implémenter une computation abstraite. C’est juste pour vous donner un exemple, ce n’est même pas proche de la réalité. Mais on reprend juste le principe et notre but c’est de vous faire comprendre le principe, et non de vous expliquer les détails techniques que nous réservons pour l’annexe consacrée au sujet [🔗](#). 🍊

L’hôte A désire envoyer un fichier contenant le message « Salut Pierrot » à l’hôte B. Au niveau de la couche transport, un mécanisme de computation de somme de contrôle est effectué et sa valeur est F00XTT01. L’hôte A a utilisé l’application ClemNet qui a pour numéro de port 56890 (au hasard) et utilise le protocole TFTP avec comme numéro de port de destination 69.



TFTP utilise UDP comme protocole de transmission.

Construisons notre segment UDP.

Nous allons ignorer la valeur du champ « longueur » (*length*) pour cet exercice. Nous y reviendrons dans le chapitre suivant, pour le moment ce n’est pas important. 🍊

Voici à quoi ressemblera notre segment UDP :

0	16
Bits	
1531	
0	56890
32	Le FooXTT01
64	Salut Pierrot

TABLE IV.4.12. – Datagramme UDP avec une somme de contrôle

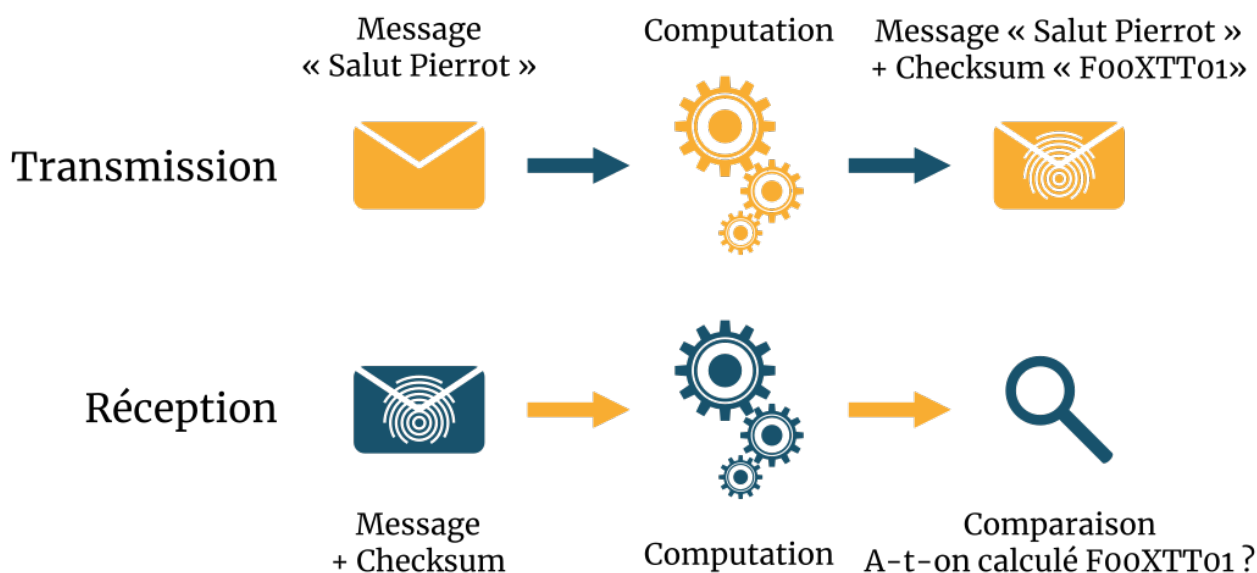
Lorsque l'hôte B va recevoir ce message, il va lui aussi calculer une somme de contrôle. La computation lui retourne la valeur F00XTT01. Il regarde alors le champ « *checksum* » (somme de contrôle) dans l'en-tête du segment reçu et compare :

Checksum reçue	Checksum calculée
F00XTT01	F00XTT01

UDP constate alors que c'est pareil, donc le message est bien intègre, aucune erreur de transmission ne s'est produite. 🍊

Par contre si les deux sommes de contrôle **ne sont pas** les mêmes, alors il y a eu une ou plusieurs erreurs de transmission entre l'instant où l'hôte A a envoyé le message, et l'instant où l'hôte B l'a reçu.

Voici un schéma pour résumer ces étapes :



zestedesavoir.com | Les réseaux de zéro

FIGURE IV.4.10. – Calcul et comparaison d'une somme de contrôle (CC BY)

Cet exemple est très, très simplifié. Le but est de vous faire comprendre le principe (d'où le titre de la sous-partie).

## Conclusion

Vous avez réussi à appréhender toutes ces notions ? Bravo, vous méritez un 20/20 ! 🍊 Si le contrôle de flux n'est pas trop difficile à comprendre quand on connaît un minimum TCP, le contrôle de congestion a de quoi déboussoler. Le principe est intéressant, après, il n'est pas indispensable de s'arracher les cheveux pour saisir le détail de chaque algorithme. C'est pour cela que nous n'en avons vu que les bases.

Nous venons aussi d'aborder le principe de la somme de contrôle. Si vous souhaitez aller plus loin, [une annexe est disponible à ce sujet](#) 🔗 . Sa lecture est facultative pour suivre le reste du cours.

## IV.5. La couche 3 : le réseau

### Introduction

Descendons une marche supplémentaire pour arriver sur la couche 3 : le réseau. Le sujet est tellement vaste que nous nous focaliserons essentiellement sur sa fonction principale : le routage.

#### IV.5.1. Rôle et matériel

##### IV.5.1.1. Le routage, qu'est-ce que c'est ?

Regardons le mot **routage** de plus près. Ça ne vous fait pas penser à **routeur** ? Ça devrait. 🍊

Le routage consiste à faire passer des données à travers des routeurs, dans le but de les faire parvenir d'un point A à un point B.

Si vous n'avez pas compris, une analogie s'impose. Supposons que vous vous trouviez à Lyon<sup>1</sup> et que vous vouliez aller à Paris en voiture, mais vous ne connaissez pas la route. Vous allumez donc votre GPS et lui demandez comment aller jusqu'à Paris. Votre appareil va effectuer une série de calculs pour déterminer la route à suivre. Quand vous serez invité à tourner à droite ou à gauche, vous le ferez, car vous ne voulez pas vous perdre. On peut comparer ces instructions au routage, car ils ont le même but : vous faire prendre la meilleure route pour arriver à bon port. Dans le détail, le fonctionnement est très différent, mais en surface, ça y ressemble. On aura tout le loisir de voir comment ça fonctionne après.

Le routage est donc l'action de router des paquets d'un réseau à un autre. Nous avons vu que lors de la transmission des paquets, les hôtes utilisaient un processus appelé ANDing, ou ET logique. Maintenant que nous sommes dans l'étude des couches du modèle OSI, il est temps d'être plus précis dans les termes. C'est le protocole IP (Internet Protocol), que nous allons étudier, qui effectue cette vérification. Souvenez-vous que chaque couche ajoute des informations en en-tête dans ce qui finit par devenir les paquets, en descendant de la couche applicative à la couche réseau. Au niveau de la couche réseau, IP doit vérifier s'il s'agit d'une communication intra-réseau ou d'une communication inter-réseau. Quelque part, c'est comme lorsque vous passez un appel. Il y a des procédures techniques au niveau de votre opérateur qui vérifient s'il s'agit d'un appel local ou international. 🍊 Par analogie, c'est ce que fait le protocole IP au niveau de la couche 3. Quand il s'agit d'une transmission en dehors du réseau, les paquets sont transmis au routeur qui se chargera du reste. Cette transmission de paquets d'un réseau à un autre est la définition de base du « routage ».

##### IV.5.1.2. Rôle de la couche

La couche réseau ou couche 3 du modèle OSI, qui correspond à la couche Internet du modèle TCP-IP, est responsable du routage. C'est même la fonction principale de cette couche. Une fois que la couche transport a assuré son rôle, les données sont envoyées à la couche réseau.

#### IV. Dans les basses couches du modèle OSI

Cette dernière se chargera d'ajouter toutes les informations en rapport avec le routage, dont notamment l'adresse IP du destinataire. C'est la seule couche du modèle OSI qui utilise la connexion logique entre hôtes. En fait, bien que cette couche ait pour rôle de déterminer le chemin physique à emprunter en se basant sur l'adresse IP du destinataire, les conditions du réseau et plusieurs autres facteurs, elle ne peut pas établir une connexion physique. Son rôle se limite à la connexion logique. Une fois qu'elle a ajouté à l'en-tête du paquet des informations qui lui sont spécifiques, ce dernier suit son cours et descend donc dans la couche 2, celle qui se chargera de liaison des données. 🍏

##### IV.5.1.3. Et le matos ?

Le matériel principal de la couche 3 est le routeur. Nous avons déjà vu ce qu'était un routeur. Il s'agit d'un matériel dont la fonction principale est d'assurer l'acheminement des paquets vers leurs destinataires en effectuant des décisions logiques déterminées par le protocole de routage utilisé.

D'autres matériels peuvent être utilisés, tels que les commutateurs avancés (*advanced switches*), aussi appelés par l'oxymore "switch de niveau 3", et les passerelles applicatives dont la fonction est de relier deux réseaux différents. Le commutateur avancé est un matériel qui fonctionne de la couche physique à la couche réseau voire transport du modèle OSI. La passerelle applicative, quant à elle, couvre toutes les 7 couches du modèle OSI, comme son nom l'indique. Elle va donc de la plus basse (couche 1, physique) à la couche applicative d'où elle tire son charmant petit nom.

Le routeur, par contre, est un matériel qui fonctionne entre les couches 1 et 3.

Voici un schéma illustrant cela :

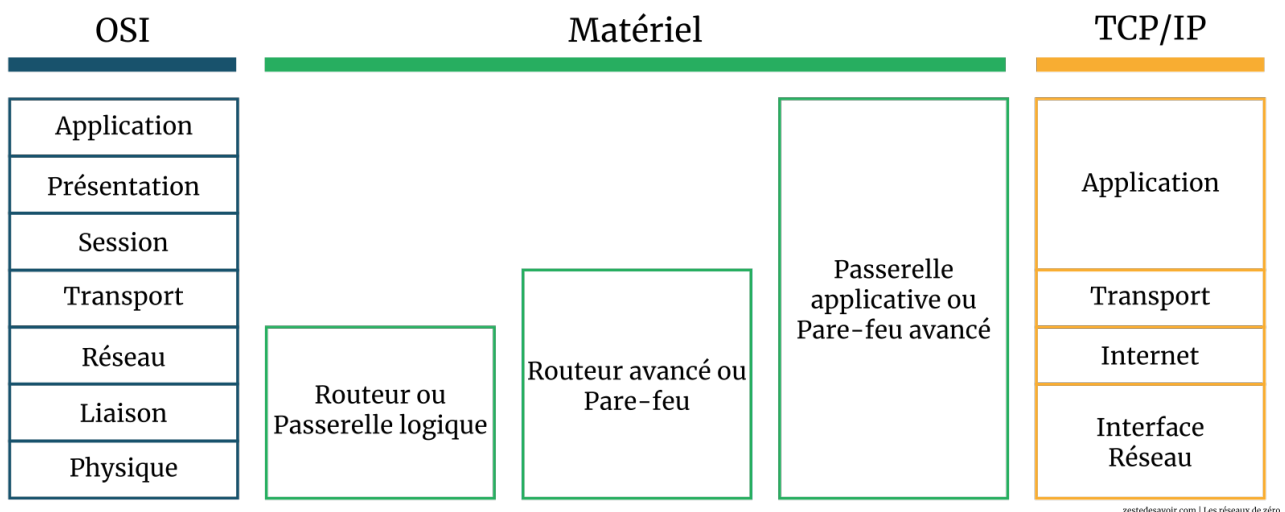


FIGURE IV.5.1. – Matériel réseau à partir de la couche 3 (CC BY)

1. Pour ceux qui ne connaissent pas bien la France, Paris est sa capitale et Lyon en est une des plus grandes villes.

## IV.5.2. Les codes de la route

### IV.5.2.1. La table de routage

Le routage est la fonction principale d'un routeur. Imaginez un réseau de milliers d'hôtes segmentés en une dizaine de sous-réseaux. Il faudrait beaucoup de routeurs pour assurer la communication entre ces 10 sous-réseaux. Les routeurs doivent s'assurer que chaque hôte de n'importe quel sous-réseau communique avec les hôtes de tous les sous-réseaux. Pour implémenter un routage effectif, il faut que les routeurs sachent prendre des décisions, pour savoir par quel routeur passer pour arriver à tel sous-réseau. Les routeurs, pour ce faire, utilisent ce qu'on appelle une **table de routage**. Quand un hôte X du réseau A veut communiquer avec un hôte Y du réseau B, les paquets seront envoyés au routeur AB qui relie le réseau A et le réseau B. Dans l'en-tête du paquet (nous allons le voir bientôt) se trouve l'adresse IP de l'émetteur et celle du destinataire. Le routeur devra donc vérifier dans sa table de routage comment faire pour arriver au sous-réseau dans lequel se trouve l'adresse IP du destinataire. Cette table de routage contient les *network ID* de tous les routeurs qui sont directement connectés au routeur AB. La table de routage contiendra également tous les chemins possibles pour atteindre un sous-réseau donné, ainsi que le coût que cela implique.

?

Ça coute cher de router un paquet ? 🍊

Oui, router un paquet implique un coût. Mais ne vous inquiétez pas, il ne s'agit pas d'argent. 🍊 En termes de routage, plusieurs facteurs déterminent le coût d'un chemin. Dans la plupart des cas, le coût est déterminé par le nombre de sauts.

?

Mais non, les routeurs ne peuvent pas sauter ! Si ? 🍊

Avec un trampoline, pourquoi pas... 🍊 Un saut, dans les termes du routage, est défini par le passage d'un paquet par un routeur. Chaque fois qu'un paquet passe par un routeur, on dit qu'il effectue un saut (*hop* en anglais).

Ainsi, dans notre exemple, le routeur AB va considérer tous les chemins qu'il a dans sa table de routage, voir quel est le chemin le moins couteux en termes de sauts et va emprunter ce dernier pour router le paquet transmis par X. 🍊

### IV.5.2.2. N'y allons pas par quatre chemins

Nous avons plusieurs fois employé le terme de chemin. C'est quoi, un chemin ? En réseau, c'est la même chose qu'un chemin dans le contexte naturel. Pour aller à l'école, au travail ou autre, il y a souvent plusieurs itinéraires possibles. Comme on dit : « tous les chemins mènent à Rome ». Cela veut dire qu'il y a plusieurs moyens pour arriver quelque part. En réseau, pour que vous visualisiez ce qu'est un chemin, nous vous avons fait un joli schéma illustratif. 🍊

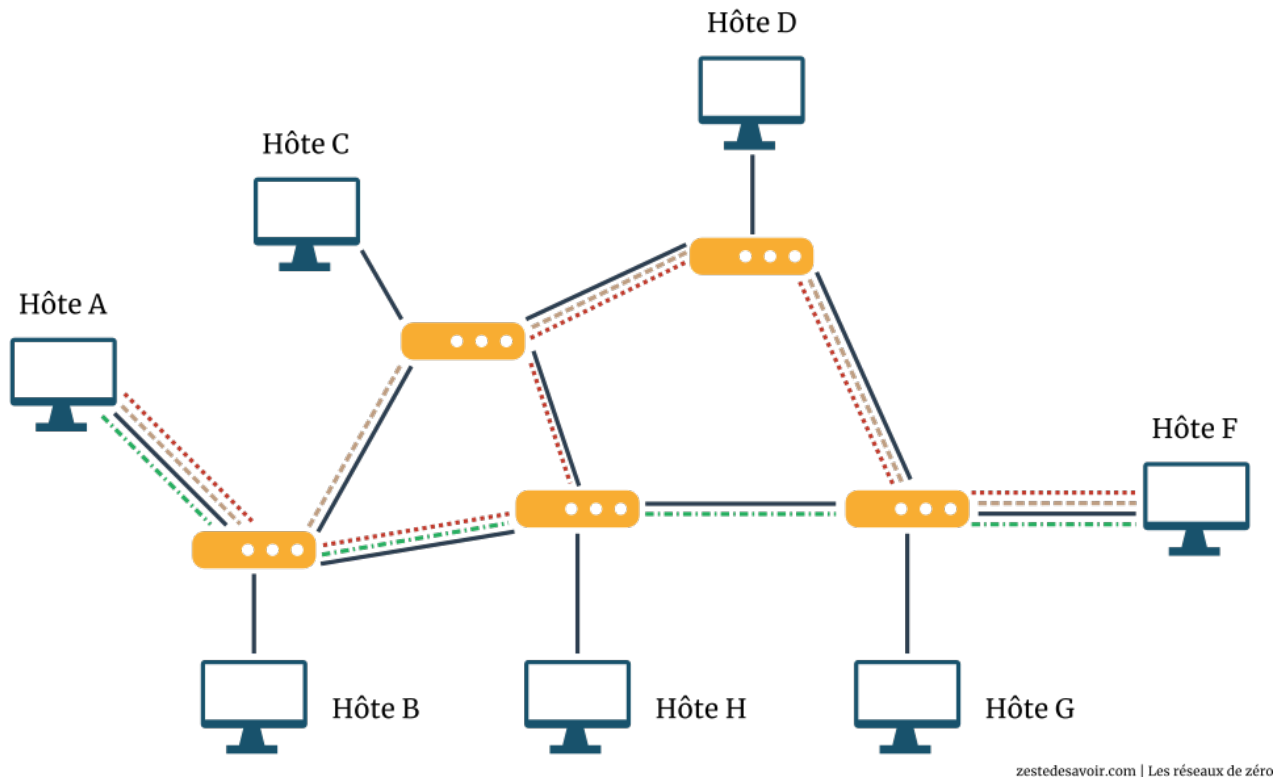


FIGURE IV.5.2. – Différents chemins sur un réseau (CC BY)

- Les traits en noir représentent les câbles qui relient les routeurs entre eux et les hôtes aux routeurs.
- Les autres traits représentent chaque chemin possible qu'un paquet allant de A à F peut suivre.

Pouvez-vous remarquer la corrélation qu'il y a entre un chemin et le nombre de saut ? En effet, plus le chemin est long, plus grand est le nombre de sauts, ce qui est logique.

- Vous pouvez voir que le chemin en vert ne coûte que 3 sauts. Il faut « traverser » 3 routeurs pour arriver à F. C'est le chemin le moins coûteux dans ce réseau.
- Le chemin en marron coûte 4 sauts, on passe par 4 routeurs avant d'atteindre F. C'est le deuxième chemin le moins coûteux.
- Par contre, le chemin en rouge est le chemin le plus coûteux. Il faut passer par 5 routeurs avant d'arriver à F.

Une table de routage contiendra toutes ces informations, et c'est par rapport à cette table qu'un routeur prendra la décision effective pour transmettre un paquet. L'intelligence qui est derrière les décisions des routeurs est fournie par les **protocoles de routage**. Nous allons aborder cette notion dans ce chapitre. 🍏

Dans cette section, nous allons jeter un coup d'œil sur la commande « route » que nous pouvons retrouver dans les systèmes Windows et Linux.



Ça sert à quoi, cette commande ?

La commande « route » sert à manipuler les tables de routage. Nous avons beaucoup parlé de tables de routage sans savoir à quoi ça ressemblait. Maintenant, c'est le moment de vérité !

🍏 Notez que cette commande nous permet en fait de faire du **routage statique**, car grâce

#### IV. Dans les basses couches du modèle OSI

à elle nous pouvons modifier les entrées de la table de routage, en ajoutant des routes, en en supprimant, etc...

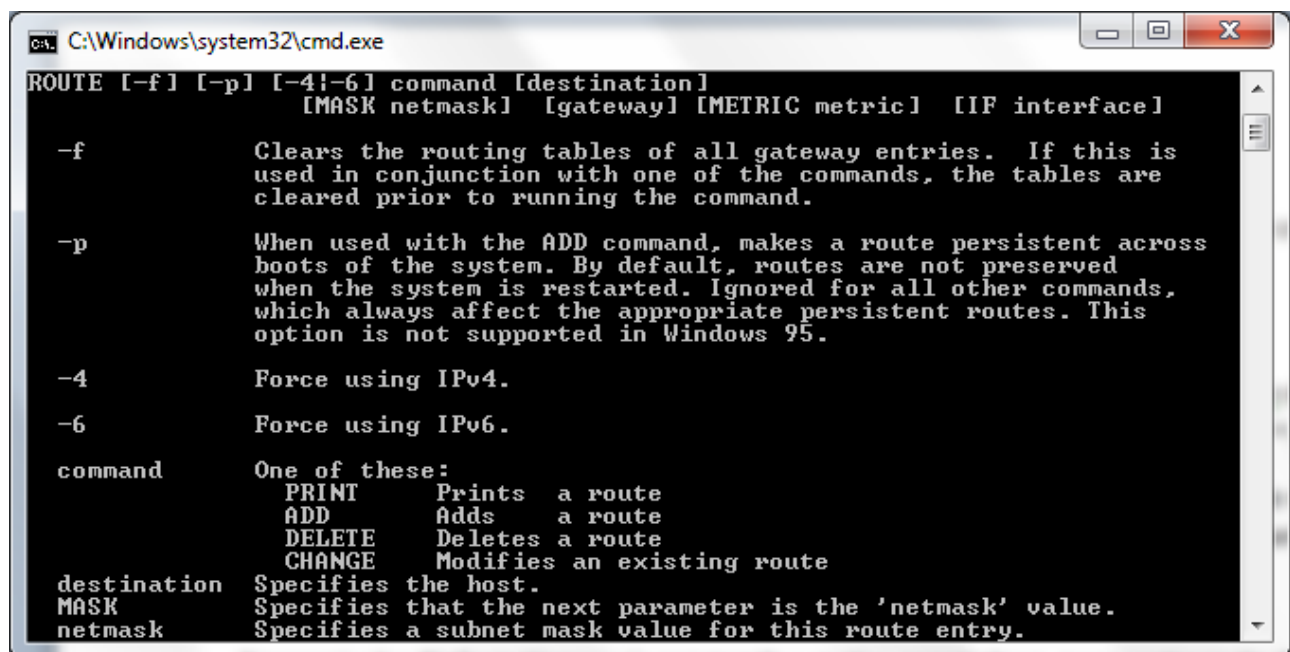
##### IV.5.2.3. À l'aide, j'ai perdu ma route !

Pour avoir plus d'informations sur la commande « route » sur Windows, ouvrez votre invite de commande et tapez `route /?`.

i

Dans cette syntaxe, on dit que `route` est la commande de base et `/?` le *switch* (rien à voir avec le matériel réseau). D'ailleurs, ce *switch* fonctionne avec toutes les autres commandes. Il sert à afficher les informations sur l'utilisation de la commande qui le précède.

Pour les Linuxiens, ouvrez votre terminal et tapez `route --help`.  
Vous obtiendrez quelque chose similaire à la capture ci-dessous :



```
C:\Windows\system32\cmd.exe
ROUTE [-f] [-p] [-4|-6] command [destination]
      [MASK netmask] [gateway] [METRIC metric] [IF interface]

-f          Clears the routing tables of all gateway entries.  If this is
            used in conjunction with one of the commands, the tables are
            cleared prior to running the command.

-p          When used with the ADD command, makes a route persistent across
            boots of the system.  By default, routes are not preserved
            when the system is restarted.  Ignored for all other commands,
            which always affect the appropriate persistent routes.  This
            option is not supported in Windows 95.

-4          Force using IPv4.

-6          Force using IPv6.

command    One of these:
            PRINT      Prints a route
            ADD        Adds a route
            DELETE     Deletes a route
            CHANGE     Modifies an existing route

destination Specifies the host.
MASK          Specifies that the next parameter is the 'netmask' value.
netmask       Specifies a subnet mask value for this route entry.
```

FIGURE IV.5.3. – Aide de la commande route sous Windows

Vous pouvez vous servir de ces informations pour apprendre à utiliser les différentes commandes. Pour ce tuto, nous allons utiliser les commandes nous permettant d'ajouter, de supprimer, de modifier une route et d'afficher le contenu de la table de routage.

##### IV.5.2.4. Afficher le contenu de la table

Vous vouliez voir à quoi ressemble une table de routage ? Sous Windows, tapez `route print` dans votre invite de commande. Sous Linux, ouvrez votre terminal et tapez `route -n`. Vous obtiendrez naturellement une table différente de la nôtre. Voici une capture de ce que nous obtenons sous Windows.

```

C:\Windows\system32\cmd.exe

=====
IPv4 Route Table
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
0.0.0.0                    0.0.0.0          10.0.0.138       10.0.0.3         25
10.0.0.0                   255.255.255.0    On-link          10.0.0.3         281
10.0.0.3                   255.255.255.255  On-link          10.0.0.3         281
10.0.0.255                 255.255.255.255  On-link          10.0.0.3         281
127.0.0.0                  255.0.0.0        On-link          127.0.0.1        306
127.0.0.1                  255.255.255.255  On-link          127.0.0.1        306
127.255.255.255           255.255.255.255  On-link          127.0.0.1        306
224.0.0.0                  240.0.0.0        On-link          127.0.0.1        306
224.0.0.0                  240.0.0.0        On-link          10.0.0.3         281
255.255.255.255           255.255.255.255  On-link          127.0.0.1        306
255.255.255.255           255.255.255.255  On-link          10.0.0.3         281
=====
Persistent Routes:
None
=====
IPv6 Route Table
=====
Active Routes:
If Metric Network Destination      Gateway

```

FIGURE IV.5.4. – Affichage d’une table de routage sous Windows

Examinons chacune des colonnes de cette fameuse table :

- **Network Destination** : cette colonne correspond au réseau ou à l’adresse de destination.
- **Netmask** : diminutif de « Network Mask », cette colonne correspond au masque du réseau. C’est le masque utilisé pour déterminer le *network ID* de l’adresse IP du destinataire.
- **Gateway** : aussi appelé « next hop » (prochain saut), cette colonne est l’adresse IP de la passerelle conduisant au réseau spécifié dans la colonne Network Destination.
- **Interface** : comme vous le savez, un routeur a plusieurs interfaces. Cette colonne spécifie quelle interface est utilisée pour transmettre les paquets dans le réseau spécifié par la première colonne.
- **Metric** : voilà notre fameuse métrique ! Il s’agit d’une valeur numérique, et comme vous pouvez le constater, les différentes entrées de la table ont des différentes valeurs de métriques. Quand plusieurs chemins conduisent au même réseau de destination, le routeur se base sur le chemin qui a la plus petite valeur dans la colonne métrique de sa table. 🍊

#### IV.5.2.5. Les différents types de routes

Nous pouvons avoir plusieurs types de route dans une table de routage et nous allons essayer de voir en quoi elles consistent.

##### IV.5.2.5.1. Network ID d’un réseau distant

Il s’agit ici de l’ID d’un réseau qui n’est pas directement lié au routeur, d’où son appellation réseau distant. Il peut alors être atteint via les autres routeurs voisins. Dans ce genre d’entrée, la colonne **passerelle** de la table de routage correspondra au routeur local par lequel il faudra passer (notre prochain saut).

#### IV.5.2.5.2. Network ID d'un réseau voisin

Il s'agit de l'ID d'un réseau qui est directement lié au routeur local. La colonne Gateway de la table correspondra alors à l'adresse IP de l'interface du routeur qui est liée à ce réseau avoisinant. Dans l'illustration ci-dessous, B est une route directe pour A. C et D sont des routes distantes que A peut atteindre via B.

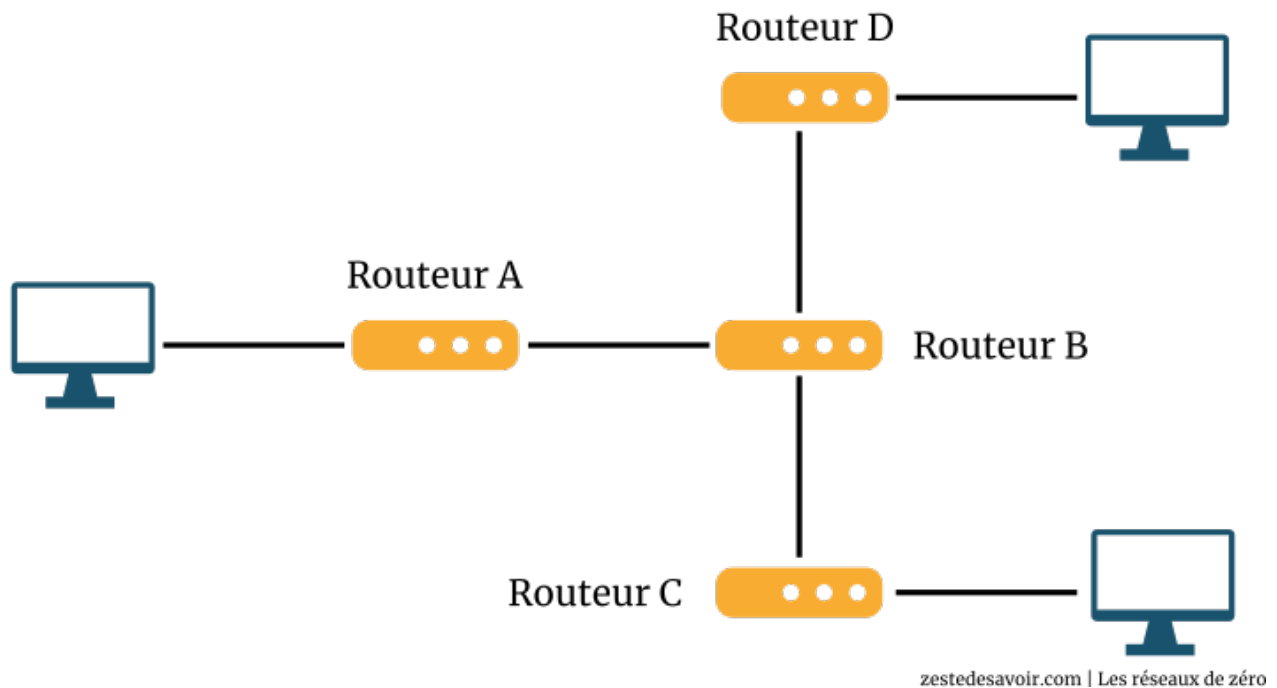


FIGURE IV.5.5. – Schéma illustrant une route distante et une route directe (CC BY)

#### IV.5.2.5.3. Route-hôte (host route)

Il s'agit d'une route vers une adresse IP précise, ce qui permet d'effectuer un genre de routage plus direct. Pour ce type de route, la colonne Network Destination est l'adresse IP de l'hôte destinataire, et la colonne Netmask aura pour valeur numérique 255.255.255.255.

?

255.255.255.255 ? Mais c'est un broadcast, non ?

Quelle bonne mémoire. 🍎 Si c'était une adresse IP, ce serait ça, mais là, on parle d'un masque. Dans celui-ci, tous les bits sont allumés, ce qui signifie que le réseau de destination ne comporte que l'adresse indiquée. En effet, tous les bits doivent être rigoureusement identiques à l'adresse fournie pour correspondre au cas, ce qui signifie donc que seul l'hôte ayant cette adresse est concerné par l'entrée de la table de routage.

#### IV.5.2.5.4. Route persistante

Toutes les routes que vous ajoutez manuellement dans votre système sont supprimées lorsque vous redémarrez votre ordinateur. Si vous voulez qu'une route « persiste » il faudrait donc ajouter une route... persistante. 🍎 Pour ajouter une route persistante dans votre table de

#### IV. Dans les basses couches du modèle OSI

routage sous Windows, il suffit d'ajouter le *switch* `-p` à la fin d'un ajout statique. En voici un exemple :

```
route add 192.161.88.0 mask 255.255.255.0 192.160.36.1 metric 20 if 1 -p
```

Sous Linux, cela dépend des distributions. Une méthode pour les systèmes Debian consiste à ajouter la commande précédée du mot-clé "up" à la fin du fichier `/etc/network/interfaces`. Exemple :

```
up route add -net 172.17.250.0/24 gw 172.17.10.141 dev eth0
```

##### IV.5.2.5.5. Route par défaut

En toute logique, c'est la route qui est utilisée lorsque la table de routage ne comprend aucune route valable vers le réseau de destination. Dans ce type de route, les colonnes Network Destination et Netmask auront une valeur numérique de 0.0.0.0. En théorie, n'importe quel réseau destination devrait fonctionner avec un masque 0.0.0.0. En effet, un ANDing avec un tel masque nous donne toujours le même résultat : n'importe quel bit peut prendre n'importe quelle valeur.

### IV.5.3. Quand utilise-t-on la passerelle ?

Nous allons à présent étudier l'algorithme de sélection de route. Nous avons déjà vu ce qu'était le routage et les chemins dans une table de routage. Il nous faut maintenant étudier comment le protocole IP sélectionne une route dans une table lors de la transmission des données entre un hôte A et un hôte B. Pour ce faire, nous allons reprendre notre schéma illustrant les différents chemins. Ce sera notre étude de cas. 🍊



C'est quoi un algorithme ?

Pour faire simple, un algorithme est une suite d'instructions précises et ordonnées conduisant à l'accomplissement d'une tâche précise. Par exemple, pour faire une omelette, il vous faut casser des œufs, les battre, mettre du sel, etc. Il y a une suite d'étapes distinctes qui doivent être respectées dans un ordre précis.

Ainsi, par "algorithme de sélection de route", nous voulons simplement désigner la suite d'étapes que le protocole IP utilise pour choisir une route.

Dans la partie I du cours, plus précisément dans le chapitre « La passerelle : les bases du routage », nous avons métaphoriquement expliqué le processus de routage d'un paquet. Maintenant que vous êtes plus avancé, nous pouvons détailler ce qui se passe dans les coulisses du protocole IP.



Pour commencer, regardons à nouveau les étapes que nous avons énoncées dans ce chapitre. En résumé, nous avons dit que si un hôte A voulait communiquer avec un hôte B, grâce au ANDing, il déterminerait si son destinataire (l'hôte B en l'occurrence) était dans le même réseau que lui. Si oui, il lui envoyait les données directement, sinon il envoyait les données à une passerelle qui se chargerait de les router.

Schématiquement, ça donnait ceci :

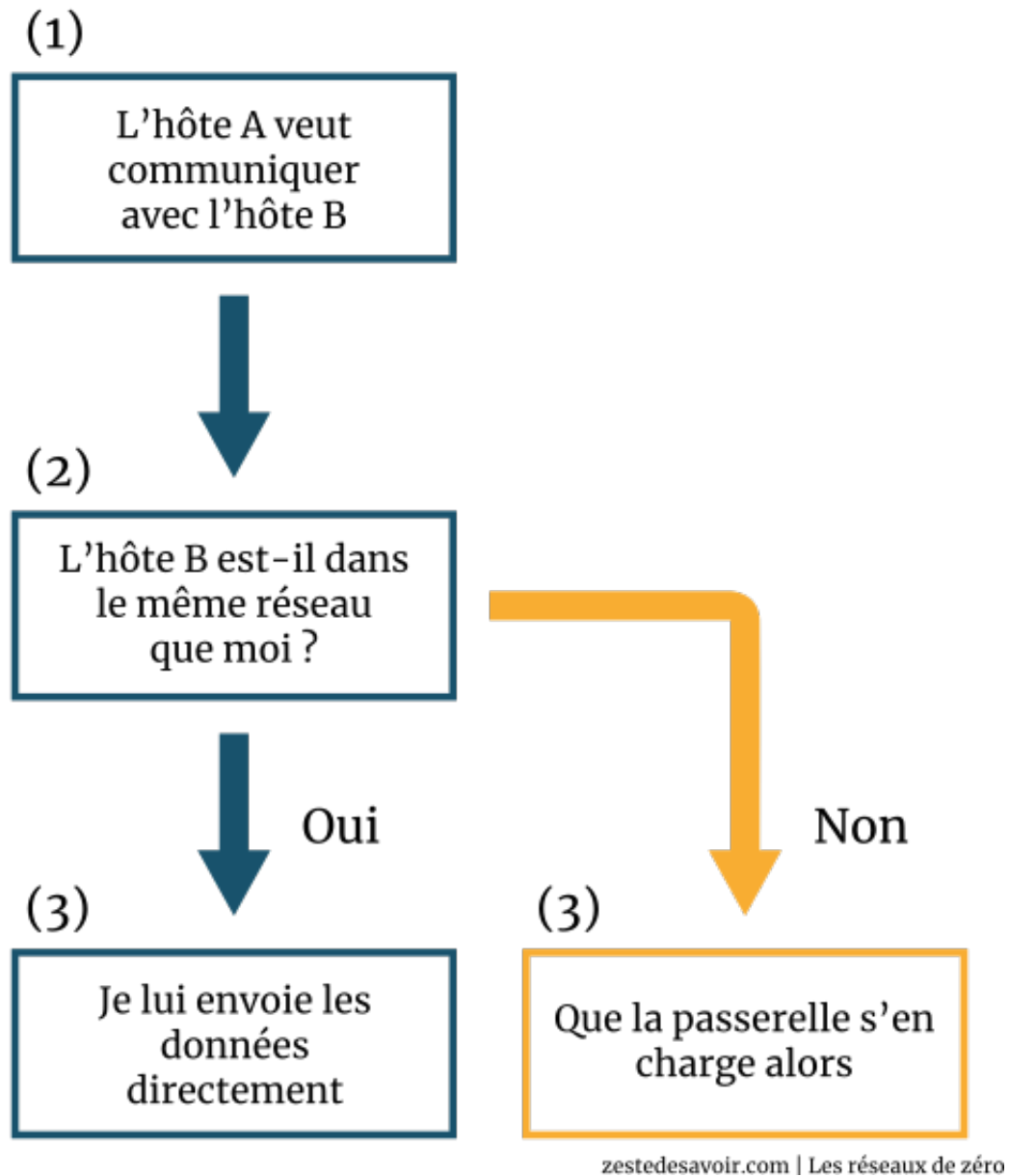
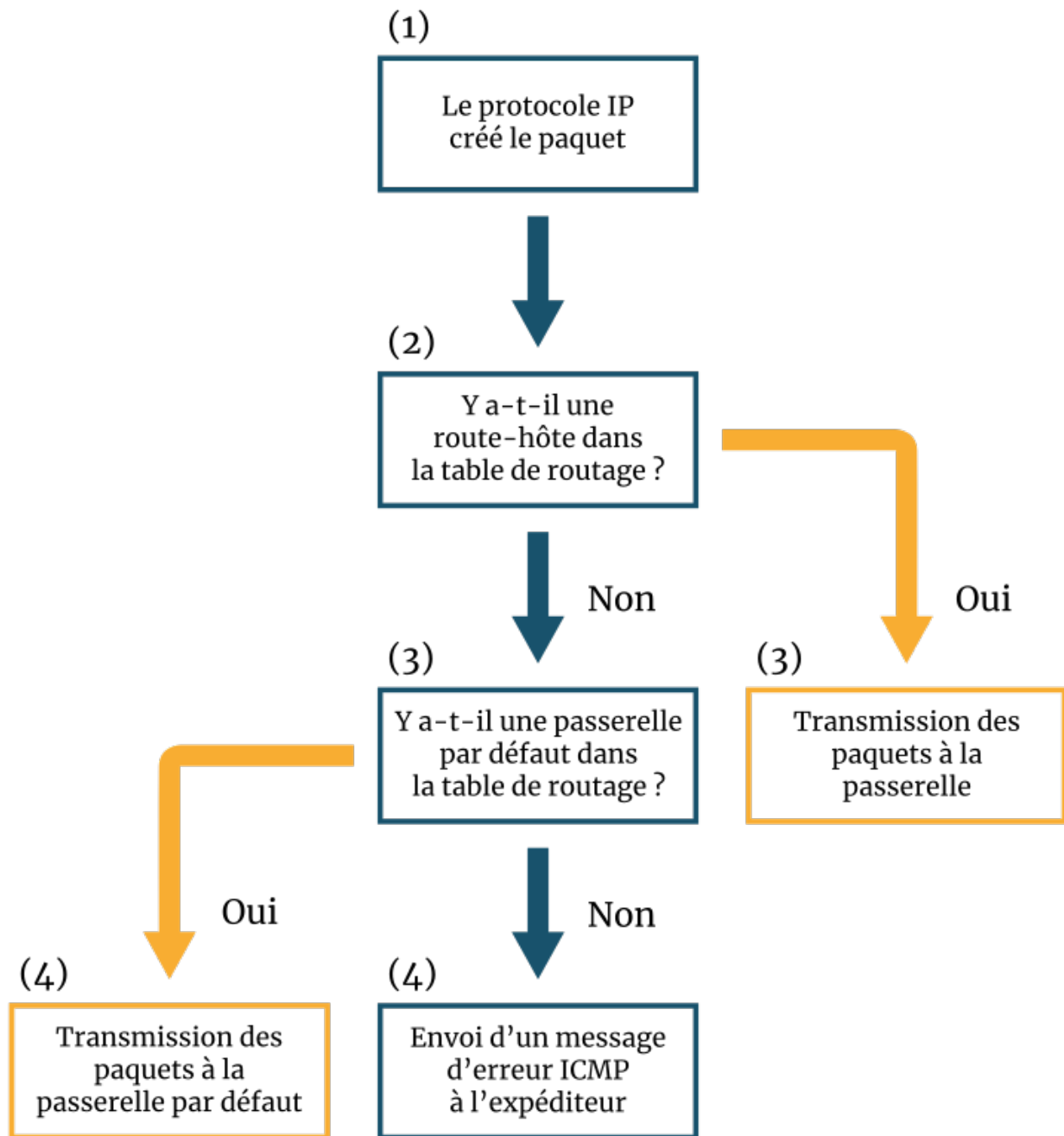


FIGURE IV.5.6. – Décision d'utiliser la passerelle ou non (CC BY)

Maintenant que vous êtes des grands, nous allons voir plusieurs autres étapes distinctes comme illustré dans le schéma ci-dessous.



zestedesavoir.com | Les réseaux de zéro

FIGURE IV.5.7. – Étapes de décision d'utilisation d'une passerelle (CC BY)

#### IV.5.3.1. La procédure de routage

Pour sélectionner une route, le protocole IP suit les étapes suivantes :

- Le protocole IP fouille les entrées de la table de routage afin de déterminer la **route-hôte** qui correspond à l'IP du destinataire. L'adresse IP se trouvera dans l'en-tête IP du paquet. Souvenez-vous que la route-hôte a l'adresse IP de destination dans la colonne Adresse Réseau et la valeur 255.255.255.255 dans la colonne du masque.
- Si le protocole IP ne trouve aucune route-hôte, il va scanner les colonnes « Adresse Réseau » et « Masque » pour chercher la route qui peut mener au réseau du destinataire. Mais que se passe-t-il s'il y a plusieurs chemins possibles pour arriver au destinataire,

#### IV. Dans les basses couches du modèle OSI

comme c'est le cas dans le schéma illustratif des chemins ? Est-ce qu'une route est choisie au hasard ? Non, c'est la route ayant le plus grand nombre de bit masqués qui sera choisie, car on cherche le cas le plus précis. 🍌 Et s'il y a deux routes ayant le même nombre de bit masqués ? La décision sera alors déterminée par la route ayant la plus petite métrique, ce qui est dépend intimement du protocole de routage utilisé.

- Si, dans la table de routage, il n'y a aucune route conduisant au destinataire, la dernière option sera alors de localiser la passerelle par défaut. Ce cas peut être intégré au point précédent, puisqu'il correspond techniquement à prendre le cas le moins précis (destination : 0.0.0.0/0), celui qui autorise tous les bits à ne pas correspondre au network ID de destination.
- Si par contre, il n'y a pas de passerelle par défaut, le paquet à ce stade est « délaissé » et une erreur ICMP est envoyée à l'émetteur. Il s'agira d'une erreur « destination inaccessible ».

Si le système a pu localiser une route conduisant au sous-réseau du destinataire, il va alors falloir déterminer l'adresse physique (MAC) du routeur qui conduira le paquet au bon endroit. Souvenez-vous que la transmission des données dans un réseau local se fait en utilisant les adresses MAC. Si le protocole IP trouve le routeur qu'il faut, cela veut dire qu'il est dans le même sous-réseau que l'émetteur, donc il faudra trouver son adresse physique ! 🍌 C'est ici qu'intervient le module ARP (Address Resolution Protocol) qui est un protocole à cheval sur les couches 2 et 3.



ARP est spécifique à IPv4. En IPv6, on utilise Neighbor Discovery Protocol (NDP).

Une table ARP contient une correspondance entre les IP et leurs adresses physiques respectives. Plus précisément, elle définit à quelle adresse physique la couche 2 doit s'adresser pour joindre une adresse IP donnée.

Voici à quoi peut ressembler une table ARP sous Windows. Vous pouvez essayer vous-mêmes avec la commande `arp -a` (valable pour Windows et Linux).

1	Interface : 192.168.1.14 --- 0x13		
2	Adresse Internet	Adresse physique	Type
3	192.168.1.1	c0-d3-8e-93-10-5c	dynamique
4	192.168.1.255	ff-ff-ff-ff-ff-ff	statique
5	224.0.0.2	01-00-5e-00-00-02	statique
6	224.0.0.22	01-00-5e-00-00-16	statique
7	224.0.0.251	01-00-5e-00-00-fb	statique
8	224.0.0.252	01-00-5e-00-00-fc	statique
9	239.255.255.250	01-00-5e-7f-ff-fa	statique
10	255.255.255.255	ff-ff-ff-ff-ff-ff	statique

Pour IPv6, on parle plutôt de **table de voisinage**. On peut la visualiser avec la commande `netsh int ipv6 show neigh` sous Windows ou `ip -6 neigh show` sous Linux.

1	Interface 19 : Ethernet	
2		
3		
4	Adresse Internet	Adresse physique
	Type	
5	-----	-----
	-----	
6	fe80::b2b2:8fff:fe73:c05a	c0-d3-8e-93-10-5c
	Joignable (Routeur)	
7	ff02::1	33-33-00-00-00-01
	Permanent	
8	ff02::2	33-33-00-00-00-02
	Permanent	
9	ff02::16	33-33-00-00-00-16
	Permanent	
10	ff02::fb	33-33-00-00-00-fb
	Permanent	
11	ff02::1:2	33-33-00-01-00-02
	Permanent	
12	ff02::1:3	33-33-00-01-00-03
	Permanent	
13	ff02::1:ff43:b83e	33-33-ff-43-b8-3e
	Permanent	
14	ff02::1:ff73:c05a	33-33-ff-73-c0-5a
	Permanent	
15	ff02::1:ffde:aa0d	33-33-ff-de-aa-0d
	Permanent	
16	ff05::c	33-33-00-00-00-0c
	Permanent	

Vous remarquerez des adresses un peu étranges, qui commencent par exemple par 224 ou ff02. Elles sont utilisées pour des types de communication particuliers que nous allons voir maintenant.

#### IV.5.4. C'est quoi, ton type de routage ?

Il existe 4 types majeurs de routage ou méthodologies de routage : unicast, multicast, broadcast et anycast. Vous avez probablement déjà entendu ces termes. Nous allons aborder chacun d'eux.

##### IV.5.4.1. Unicast: un seul destinataire

L'unicast consiste à transmettre les paquets à **un seul destinataire** (*uni* comme unique, un). Un exemple de routage unicast est lorsque vous visitez une page web toute bête, du genre [perdu.com](http://perdu.com) [↗](#). Pas de grosse infrastructure, on est à peu près certain que la requête ne peut aller que d'un client à un serveur précis.

#### IV.5.4.2. Multicast: restriction à un groupe

Le multicast, c'est un peu comme effectuer plusieurs unicast à un groupe déterminé, mais en n'utilisant qu'une seule adresse. Sur un réseau, différents hôtes peuvent s'abonner ou se retirer d'un groupe pour recevoir des données sans que l'émetteur n'ait quoi que ce soit à changer. Cela a diverses applications, comme la transmission d'informations de routage ou encore la diffusion de flux multimédia. Les récepteurs sont alors sous une même adresse IP multicast. L'adresse logique utilisée dépend intimement du protocole utilisé. Dans le cas de protocoles de routage, RIPv2 utilisera l'adresse 224.0.0.9 pour multicaster des paquets, tandis qu'OSPF utilisera l'adresse 224.0.0.5 pour envoyer de paquets de signalisation « Hello » aux routeurs du réseau physique.



En IPv4, les adresses multicast sont forcément de classe D. En IPv6, elles commencent par le préfixe ff00::/8.

Voici [une liste de quelques adresses multicast](#) et leur utilisation.

#### IV.5.4.3. Le broadcast

Le broadcast, c'est simple : on envoie à tout le monde. 🍊 Bon, en vrai, il y a quelques subtilités. On peut envoyer des paquets en broadcast sur son réseau logique. Nous avons vu lors des chapitres sur l'adressage que, conventionnellement, la dernière adresse IP d'un sous-réseau était son adresse de broadcast. On peut donc l'utiliser pour communiquer avec tous les hôtes de son propre sous-réseau.

Mais il y a une autre possibilité. On peut utiliser l'adresse 255.255.255.255. Avec celle-là, vous pouvez contacter tout le monde. Enfin presque. Si ça se retrouvait sur Internet, ce serait un sacré bordel. 🍊 Les routeurs ne laissent pas passer ces broadcasts, mais cette adresse permet d'arroser tous les hôtes de votre réseau **physique**. Vous vous souvenez de la différence entre un réseau logique et physique ? Dans le cas présent, vous pouvez contacter un hôte dans un réseau logique différent du vôtre s'il est dans le même réseau physique.



Et en IPv6 ?

Au risque de vous surprendre, ce n'est pas possible. Le broadcast n'existe pas en IPv6. 🙌

#### IV.5.4.4. Anycast: à n'importe qui ?

Nous finissons par le plus compliqué. Le préfixe « any », dans *anycast*, est un mot anglais qui signifie « n'importe ». On pourrait croire, à première vue, que ce type de routage consiste à router des paquets à n'importe qui, ce qui n'aurait aucun sens. Vous ne pianoteriez pas un numéro au pif sur votre téléphone, sans savoir à qui il est ni même si le numéro est attribué (ou alors vous êtes sacrément tordu-e 🍊 ).

Le principe de l'anycast, c'est de router des paquets au destinataire le plus proche lorsqu'il existe plusieurs chemins conduisant au même réseau. Si trois routeurs B, C, D conduisent tous les trois au routeur E, un routeur A enverra alors les paquets en anycast. Ainsi le plus proche de ces 3 routeurs recevra les paquets et les acheminera au destinataire final. Le routeur le « plus proche » est déterminé par le protocole qui est utilisé. Dans le cas de RIP qui mesure la

distance par le nombre de sauts, le routeur le plus proche serait celui qui nécessite le plus petit nombre de saut.

Quand on effectue ce genre de routage, on dit que l'on « anycast » (du verbe « anycaster » 🍌) les paquets. C'est pas dit que le dictionnaire soit très d'accord, donc ça reste entre nous !

##### IV.5.4.4.1. Un exemple de l'application de ce plan de routage

Vous ne voyez pas encore les avantages de l'anycast ? Essayons d'illustrer cela.

Nous allons décrire une architecture possible de Zeste de Savoir. Ce n'est pas celle actuellement en place, mais elle permet de montrer les avantages de l'anycast.

Zeste de Savoir, à en croire les stats, compte 10.310 membres au moment de la rédaction de ces lignes. Heureusement que ces derniers ne se connectent pas tous au même moment, cela augmenterait considérablement le nombre de requêtes que le serveur principal doit recevoir. Nous allons supposer que l'architecture de ZdS est comme suit :

- 2 serveurs de contenus : nous allons considérer que le site a des milliers de contenus (cours, articles, billets, ...) disponibles. ZdS décide donc d'avoir deux serveurs : un serveur principal « normal » qui héberge les cours et autres, et un autre serveur « miroir » qui est une copie conforme de tous les contenus hébergés par le premier. Appelons le serveur principal « contenus\_serveur » et le miroir « contenus\_bis ».
- 1 serveur de membres : ce dernier appelé « membres\_serveur » héberge tous les membres du site et leurs informations associées (pseudo, mot de passe, signature, avatar, etc.), sous forme d'une base de données.
- 2 serveurs d'images : rien qu'à voir le nombre d'images par tutoriel, sur le forum, etc., on va supposer que ZdS a déployé deux serveurs d'images, le deuxième servant de miroir. Nous allons les appeler « images\_serveur » et « images\_bis ».
- 2 serveurs web : ces deux serveurs hébergent toutes les pages web du site. C'est donc le serveur qui se charge de vous afficher les pages selon vos requêtes HTTP envoyées via votre navigateur. Appelons les « zds » et « zds\_bis ».
- 1 serveur front-end : ce serveur n'héberge pas les données du site, il sert de gestionnaire de requête. C'est ce dernier qui va intercepter vos requêtes et les transmettre aux autres serveurs.

Voici un schéma de l'architecture décrite.

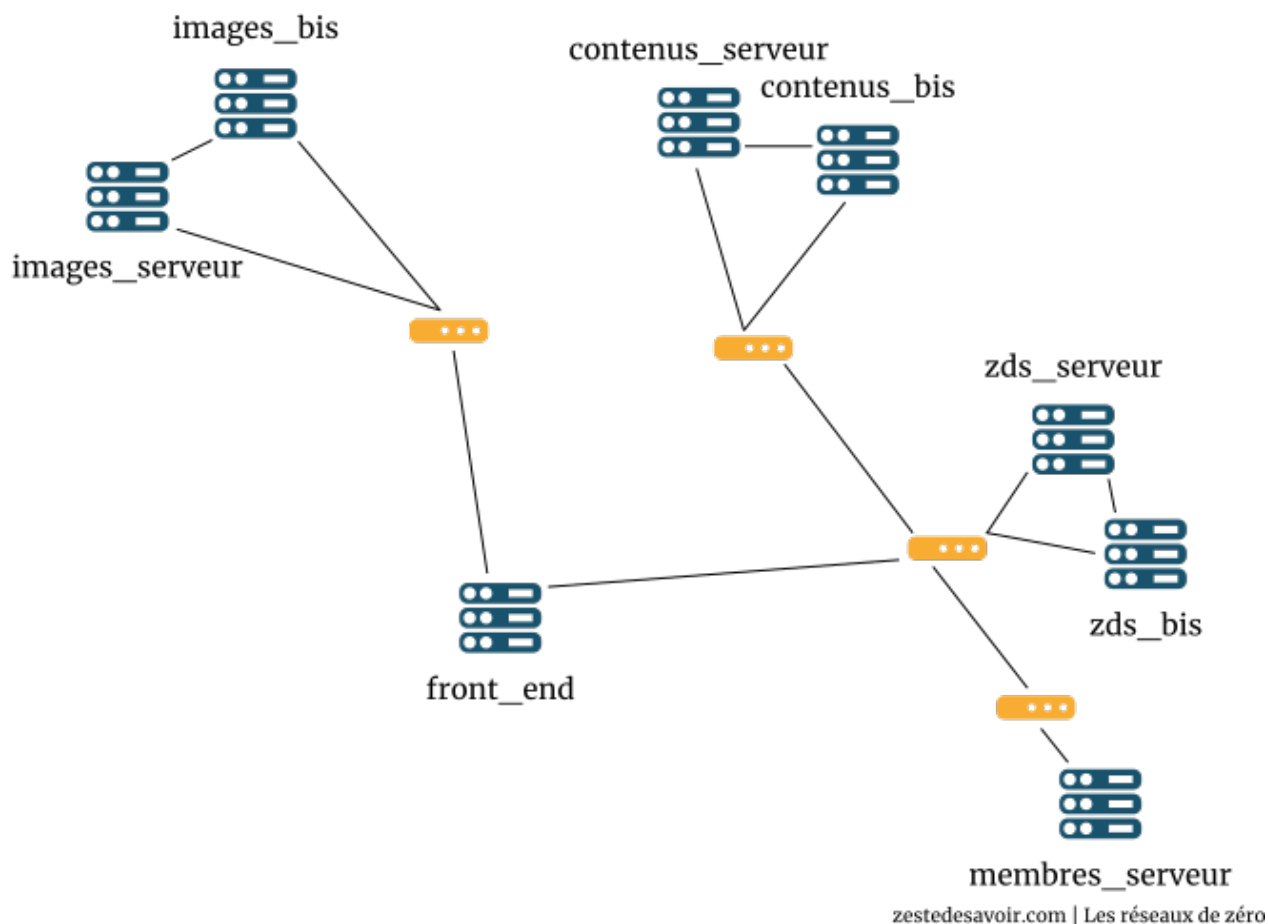


FIGURE IV.5.8. – Architecture imaginaire de Zeste de Savoir (CC BY)

i

L'anycast a de l'intérêt dans des grands réseaux avec un certain nombre de routeurs. On va considérer que les serveurs de ZdS sont dispatchés dans un plus grand ensemble et que les liens ne sont pas directs.

Que se passerait-il si tous les membres essayaient de se connecter au même moment ? Cela ferait plus de 10.000 requêtes à gérer simultanément par ressource demandée ! 🍊 Les serveurs sont peut-être assez robustes pour faire cela, néanmoins, on peut minimiser la charge en décentralisant les ressources. Par exemple, on a 2 serveurs pour les images. On peut les grouper sous une adresse anycast logique. En d'autres termes, de « l'extérieur » on ne verra qu'un seul serveur d'images (une seule adresse logique, adresse d'anycast). Cependant, sous cette adresse, il y aura 2 serveurs distincts auxquels on peut anycaster les paquets. 🍊

Ainsi, lorsque ZdS aura à gérer 10.000 requêtes vers des images de façon simultanée, le serveur front-end diffusera les requêtes en les anycant au serveur d'images le plus proche en termes de « distance ». La distance peut être déterminée par le nombre de sauts, si c'est le protocole RIP qui est utilisé.

Cette nouvelle architecture va optimiser la gestion des requêtes et la performance des serveurs.

Un protocole pratique comme EIGRP fait encore mieux : il peut considérer également la saturation d'un lien pour déterminer à quel serveur envoyer les requêtes. Ainsi, les deux serveurs d'images pourront se partager les tâches, au lieu de laisser un seul serveur faire tout le boulot.



L'implémentation d'un plan de diffusion anycast dans une partie du réseau est assez complexe et prend plusieurs autres paramètres en considération. Si le plan de routage est mal fait, il y aura logiquement des sérieux problèmes de communication. Nous n'allons pas rentrer dans les détails techniques des contraintes de l'anycast, ça fait déjà beaucoup à intégrer pour le moment ! 🍊 Nous avons commencé à évoquer des protocoles de routage. Il est maintenant temps d'introduire proprement cette notion.

### IV.5.5. Introduction aux protocoles de routage

Commençons par préciser qu'il y a deux manières de faire du routage : le routage statique et le routage dynamique.

#### IV.5.5.1. Routage statique

Les routeurs dans un réseau doivent communiquer normalement et s'échanger des informations telles que le contenu de la table de routage, que nous avons déjà évoquée. Le processus d'échange d'informations entre routeurs dépend du genre de routage utilisé. Dans un routage statique, c'est vous, l'administrateur réseau, qui devez construire et mettre à jour manuellement vos tables de routage. Donc dans un routage statique, le contenu des tables de routage est également statique. Ce genre de routage n'est pas vraiment pratique pour des raisons évidentes :

- Les routeurs ne découvriront pas automatiquement les Network ID des autres réseaux, ce sera à vous de les leur apprendre par une configuration manuelle ;
- Les routeurs ne pourront pas communiquer entre eux pour s'échanger des informations, ce sera à vous de modifier les changements des données de votre réseau dans chaque routeur, manuellement ;
- Les routeurs ne seront pas intelligents et pourront garder des données erronées dans leur table. Étant donné qu'ils ne communiquent pas entre eux dans un routage statique, cela veut dire que si un chemin vers un réseau n'est plus praticable, un routeur continuera à considérer ce chemin comme étant valable, du moins, tant que vous ne l'aurez pas changé.

En bref, le routage statique, cela signifie que le routage est défini manuellement par l'administrateur.

#### IV.5.5.2. Routage dynamique

Le routage dynamique est exactement le contraire du routage statique. 🍊 Tout se fait automatiquement grâce à un protocole de routage. Cette section vous servira d'introduction aux protocoles de routage. Le chapitre qui suit sera dédié à ces derniers, il est donc important de poser une bonne base de connaissance avant de s'y attaquer. 🍊 Introduisons d'abord ce concept.

Il est possible d'avoir un réseau composé de plusieurs dizaines de routeurs. Ces derniers doivent constamment communiquer en s'échangeant des informations sur les routes. Comme vous le savez certainement, en réseau ce sont des protocoles qui permettent la communication entre les hôtes. Le rôle d'un protocole de routage consiste donc à définir les règles et principes de communication entre routeurs, pour ce qui concerne le seul routage.

En résumé, un protocole de routage, c'est l'intelligence qui régit la manière dont les routeurs communiquent entre eux pour nous offrir le meilleur service de routage possible, c'est-à-dire le moins coûteux, le plus rapide, le plus pratique.

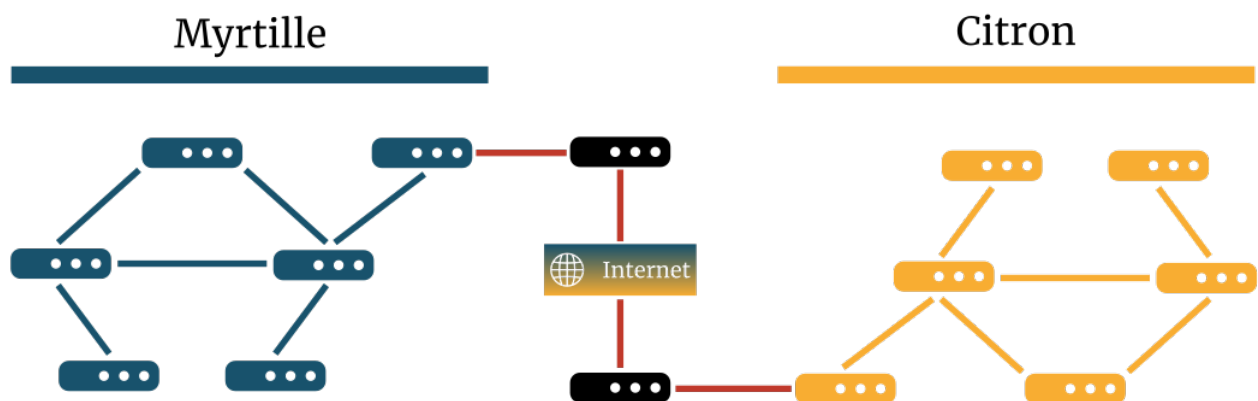
#### IV.5.5.2.1. Les différents protocoles de routage

Il existe de nombreux protocoles de routage. Un routeur peut supporter plusieurs protocoles en même temps. Pourquoi en avoir plusieurs, s'ils servent tous à faire la même chose ? La réponse se trouve dans la méthodologie. Ces protocoles font certes la même chose, mais certains sont plus pratiques que d'autres, d'autres ont des contraintes plus couteuses par exemple. Citons pêle-mêle les protocoles les plus célèbres : RIPv1 et RIPv2, OSPF, EIGRP, IGRP, BGP, IS-IS.

Ces protocoles peuvent se classer dans des « familles ». Il en existe deux :

- **IGP** (Interior Gateway Protocol, protocole de routage interne) : c'est la famille des protocoles qui peuvent échanger des informations de routage avec des **systèmes autonomes** (AS, *Autonomous Systems*). Il s'agit, pour faire simple, d'un ensemble de réseaux IP contrôlés par une organisation ou une entreprise.
- **EGP** (Exterior Gateway Protocol, protocole de routage externe) : c'est la famille des protocoles qui déterminent la disponibilité d'un réseau entre deux systèmes autonomes. Il s'agit des protocoles permettant le routage entre deux systèmes autonomes différents. Les fournisseurs d'accès à Internet, par exemple, utilisent un protocole de cette famille pour effectuer un routage externe. C'est la famille du principal protocole de routage utilisé par Internet, **BGP** (Border Gateway Protocol).

Voici un schéma illustrant quel type de protocoles est utilisé à quel moment. Nous avons schématisé un réseau constitué de deux systèmes autonomes (Myrtille et Citron).



zestedesavoir.com | Les réseaux de zéro

FIGURE IV.5.9. – Systèmes autonomes faisant appel à des protocoles de routage (CC BY)

Les traits centraux en rouge représentent les protocoles de la famille EGP, ceux qui se chargent du routage sur Internet. En bleu et jaune, sur les côtés, vous avez des protocoles de type IGP, pour le routage interne.

Vous arrivez à suivre ? Dans un futur chapitre, nous allons étudier quelques protocoles de routage. 🍊

## Conclusion

Que d'aventures ! La couche 3 est un domaine d'étude conséquent, et dans ce tutoriel, nous n'en voyons qu'une partie. Dans ce chapitre, nous avons posé les bases du routage, ce processus d'acheminement des données au travers des réseaux. Nous avons vu ce qu'était une route, dans quels cas nous avons besoin d'une passerelle, les possibilités d'adressage en fonction de qui on souhaite contacter, et enfin, nous avons introduit la notion de protocole de routage.

#### *IV. Dans les basses couches du modèle OSI*

Le chapitre suivant est beaucoup plus léger et amusant. Nous allons illustrer le routage au travers d'exemples sans rentrer dans la technique. Profitez-en bien ! 🍊

## IV.6. Le routage par l'exemple

### Introduction

Précédemment, nous vous vu ce qu'est le routage et comment se prenaient les décisions d'aller par tel ou tel chemin. Pour consolider cette notion, ce chapitre servira à illustrer ce concept. Le routage est intéressant à étudier quand on a de grands réseaux. Ben oui, quel intérêt de parler de routage sur un réseau 192.168.0.0/24 et un réseau 192.168.1.0/24 ? Il est beaucoup plus instructif de parler de routage sur des réseaux de grande envergure. Nous allons prendre deux exemples pour illustrer cette section : un réseau privé d'une très grande compagnie que nous appellerons Agrume, et le réseau public par excellence, Internet.

#### IV.6.1. En route, mauvaises troupes !

Agrume est une multinationale qui fabrique des boissons à base de fruits. Son siège est à Paris. Là-bas, il y a tout plein de bureaux pour les gens du marketing, pour les cadres, pour le support informatique, etc. Mais à Paris, l'espace est limité et l'immobilier est hors de prix. Les usines, nécessitant beaucoup de place, sont à Reims, une ville française célèbre pour son champagne. Pour pouvoir superviser ce qui se passe à Reims, les usines sont connectées avec le siège, à Paris. De plus, pour pouvoir vendre ses boissons plus facilement en Amérique du Nord, Agrume dispose d'autres usines à Montréal, au Canada. Elles aussi ont besoin d'être connectées au siège. Pour davantage de sécurité, chaque site dispose de réseaux segmentés, ayant chacun ses propres protections. Ainsi, à Paris, les gens du marketing ne peuvent pas accéder aux réseaux des usines, mais peuvent communiquer avec le support informatique. Les cadres, en revanche, sont sur un réseau depuis lequel ils peuvent accéder à tous les autres. Définissons le plan d'adressage suivant :

.	Cadres (Paris)	Support informatique (Paris)	Marketing (Paris)	Usine d
Réseau	75.12.14.0/24	75.12.15.0/24	75.12.16.0/24	51.3.1.0/
Adresse du routeur	75.12.14.1	75.12.15.1	75.12.16.1	51.3.1.1

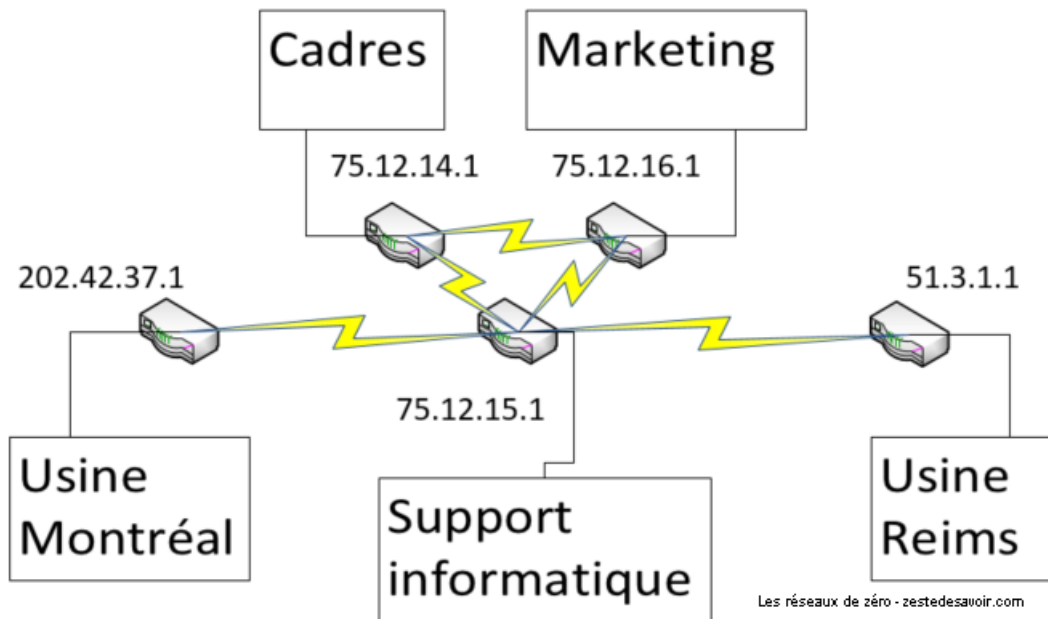


FIGURE IV.6.1. – Schéma représentant l'infrastructure du réseau Agrume

On décide arbitrairement que seul le routeur du support informatique est connecté avec celui de l'usine de Reims et celui de l'usine de Montréal. En revanche, les trois routeurs parisiens sont reliés entre eux.

?

Comment un routeur à Paris peut être relié à un autre de l'autre côté de l'Atlantique ?  
On n'a quand même pas déroulé un câble au fond de l'océan juste pour ça ? 🍊

Quand même pas. 🍊 Même pour Reims, qui est approximativement à 200 km de Paris, on ne va pas physiquement brancher un câble de chaque côté. Ce sont des opérateurs télécoms qui s'occupent de ça. Nous, on ne voit pas ce qu'il y a entre Reims, Paris et Montréal, on ne s'intéresse qu'à notre réseau interne.

Prenons un cadre, dont le poste a l'adresse 75.12.14.30, qui veut communiquer avec une usine de Montréal, dont le poste de supervision a l'adresse 202.42.37.87. Il souhaite connaître en temps réel le rythme de production d'un soda quelconque. Comment la communication va se faire ?

Si vous vous souvenez de la partie sur l'adressage, vous savez que le poste du cadre ne peut pas atteindre directement celui de supervision, car ils ne sont pas dans le même réseau. Il doit donc passer par un routeur. Généralement, les postes informatiques ont juste une passerelle par défaut qui correspond à leur routeur de sortie.

Le paquet émis par le cadre arrive donc sur son routeur de sortie. C'est là que les choses se compliquent. Par où va-t-il passer ensuite ?

?

Par le routeur du support informatique, pardi ! C'est évident !

Pas si vite ! Et si le câble entre le routeur des cadres et celui de l'informatique est hors service, ou saturé ? Eh bien, les routeurs ont la faculté de s'échanger entre eux des informations sur l'état du réseau. Ils peuvent se tenir au courant quasiment en temps réel de problèmes de liens coupés, de saturation, ... En faisant cela, ils se forment des **tables de routage**. Ces tables leur permettent de savoir quel est le meilleur chemin à prendre pour une destination donnée. En

#### IV. Dans les basses couches du modèle OSI

fonction des besoins, on configure les routeurs pour utiliser différents **protocoles de routage**. Ce sont ces protocoles qui définissent comment les routeurs s'échangent des informations et comment ils établissent ces tables de routage. Nous en étudierons quelques-uns par la suite. Voilà une belle introduction au routage sur un réseau interne. Et sur Internet alors ? Ce n'est pas tout à fait la même limonade ! 🍹

#### IV.6.2. Dans le grand bain d'Internet

Le principe de base ne change pas, seulement, on va se retrouver là sur des réseaux d'opérateurs. La partie que nous avons volontairement éludée tout à l'heure va devoir être quelque peu éclaircie.

Nous nous baserons ici sur la manière dont fonctionne l'accès à Internet résidentiel en France. Les principes du routage restent les mêmes partout. Prenons l'exemple de votre poste qui envoie des paquets à `zestedesavoir.com`. Le plus parlant est encore de visualiser par où passent ces paquets.



Visualiser ? Comment ?

Grâce à un outil formidable que vous avez certainement déjà sur votre ordinateur : **tracert** ! 🍹 Ce programme détermine le chemin parcouru, en termes de routage, par les paquets IP. Faisons tout de suite un test. Pour cela, ouvrez votre invite de commande comme vous l'aviez fait pour voir votre adresse IP il y a de cela un moment, et entrez la commande suivante : `tracert zestedesavoir.com` pour Windows, ou bien `traceroute zestedesavoir.com` sous Linux et Mac OS.

Si vous êtes sur smartphone, il y a des applications pour cela. Vous pouvez aussi le faire depuis des outils en ligne, mais vous n'aurez pas le chemin depuis votre appareil.

Et devant vos yeux ébahis défilent des lignes qui semblent venir tout droit d'une autre planète !



```
1 Détermination de l'itinéraire vers zestedesavoir.com [92.243.7.44]
2 avec un maximum de 30 sauts :
3
4 1    <1 ms    <1 ms    <1 ms    livebox.home [192.168.1.1]
5 2    2 ms     1 ms     1 ms     80.10.235.17
6 3    1 ms     <1 ms    2 ms
   ae112-0.nclyo201.Lyon3eArrondissement.francetelecom.net
   [193.253.87.254]
7 4    2 ms     2 ms     2 ms
   ae42-0.nrlyo201.Lyon3eArrondissement.francetelecom.net
   [193.252.101.234]
8 5    2 ms     1 ms     2 ms
   ae43-0.nolyo101.Lyon3eArrondissement.francetelecom.net
   [193.252.101.225]
9 6    2 ms     2 ms     2 ms    193.252.227.98
10 7    7 ms     8 ms     8 ms    lag-ly-1.th2-1.rt.hopus.net
    [37.77.32.6]
```

#### IV. Dans les basses couches du modèle OSI

11	8	8 ms	8 ms	7 ms	lag-th2-1.pa3-2.rt.hopus.net [37.77.32.11]
12	9	9 ms	13 ms	12 ms	gandi.std-1.rt.hopus.net [37.77.38.5]
13	10	8 ms	8 ms	8 ms	abr-core4-gw.gandi.net [217.70.176.157]
14	11	11 ms	10 ms	10 ms	i0.edge-b.sd5.paris.gandi.net [217.70.176.11]
15	12	10 ms	9 ms	13 ms	xvm-136-5.sd5.ghst.net [155.133.136.5]
16	13	10 ms	10 ms	10 ms	zdsprod1.zestedesavoir.com [92.243.7.44]
17					
18	Itinéraire déterminé.				

i

Pour avoir un retour plus visuel et plus parlant, l'opération a été forcée sur IPv4.

Déchiffrons ces informations, qui, contrairement aux apparences, ne signifient pas que notre planète va être colonisée par d'étranges monstres de Saturne. 🍊

La première ligne correspond au routeur que nous utilisons, ici, une Livebox. La seconde correspond au **BAS** auquel notre routeur est relié. Ce serveur est une porte d'entrée sur Internet et se situe dans les locaux de notre opérateur télécom. Les lignes suivantes désignent d'autres routeurs appartenant à notre opérateur, puis passent par d'autres réseaux dont celui de l'hébergeur Gandi, avant d'arriver sur le serveur que nous souhaitons joindre. Il est possible que certains routeurs soient physiquement directement reliés entre eux, vraisemblablement par une fibre optique, mais rien ne nous le garantit. Les opérateurs peuvent dissimuler certains équipements pour un tas de raisons que nous n'évoquerons pas ici, car ce n'est pas le sujet.

i

Grâce aux noms de domaine associés aux routeurs, on peut lire que notre requête part de Lyon et passe par Paris. Des outils en ligne permettent de visualiser sur une carte ce genre de chemin. Certains routeurs peuvent refuser de répondre : dans ce cas, la commande affiche des étoiles (\*).

Ça, c'était la partie amusante. 🍊 Nous espérons que cela vous permet de mieux visualiser la notion de routage et ce que cela implique.

## Conclusion

Alors, ce n'était pas si compliqué que ça, avouez ! Ah... Vous êtes jaloux de vos données et aimeriez pouvoir faire Paris - Tokyo à la vitesse de la lumière ? Et vous demanderiez votre chemin à votre *backbone* opérateur ? C'est beau de rêver. 🍊

Dans le prochain chapitre, on rentrera dans les détails : comment le routeur sait vers où router les paquets, comment il choisit la meilleure route, etc.

## IV.7. Les protocoles de routage

### Introduction

Dans ce chapitre, nous allons attirer votre attention sur les protocoles de routage. Vous avez pu lire une introduction à cette notion dans le chapitre précédent. Il est maintenant temps de voir quelles sont les caractéristiques des différents protocoles de routage. Nous allons nous focaliser sur RIP, OSPF et BGP.

#### IV.7.1. Direct au cimetière avec RIP

Tout comme les adresses MAC n'ont aucun rapport avec les ordinateurs Apple, RIP n'a rien d'une pierre tombale. Loin de signifier *Requiescat In Pace*, on parle ici de **protocole d'information de routage** (*Routing Information Protocol*). Il permet de faire du **routage dynamique**, c'est-à-dire sans intervention humaine. Les routeurs peuvent s'échanger dynamiquement les informations de routage au moyen de ce protocole.

Il est défini par la [RFC 1058](#) et utilise un algorithme de routage dit à **vecteur de distances** (*distance vector*). Retenez ce nom, nous étudierons cette notion par la suite. RIP est classé dans la famille des protocoles de routage interne ( **IGP** ), que nous avons évoquée dans notre introduction aux protocoles de routage. 🍊 Les protocoles de routage à vecteur de distances, comme RIP, sont basés sur l'[algorithme de Bellman-Ford](#) .

Il existe deux versions de ce protocole. Commençons par étudier la différence entre ces versions avant de plonger dans la technique.

##### IV.7.1.1. Première version

La toute première version de ce protocole ne pouvait fonctionner que dans un réseau utilisant l'adressage par classe. 🍊 Cela veut donc dire que dans un réseau utilisant RIPv1 comme protocole de routage, on ne peut pas utiliser des techniques telles que l'implémentation des masques de sous-réseaux à longueur variable (VLSM) ou l'agrégation des routes (*supernetting*). Une autre faiblesse de cette version est qu'elle ne peut supporter qu'un maximum de 15 sauts. Cela veut dire que s'il y a plus de 15 sauts entre deux passerelles, RIP ne pourra pas assurer son devoir de routage. Ce n'est pas tout : RIPv1 ne supporte pas l'authentification. 🍊 Ainsi, il accepte et intègre tous les messages qu'il reçoit de tout le monde sans sourciller. Cela fait donc de RIPv1 un protocole très vulnérable. 🍊 Pour terminer sur les caractéristiques de cette version, les routeurs utilisant RIPv1 mettent à jour leurs tables de routage en les *broadcastant* (diffusant) sur tous les routeurs adjacents (ou voisins).

##### IV.7.1.2. Alléluia, la v2 est là

Comme nous l'avons vu, la version 1 est ~~toute-pourrie~~ limitée. Elle était pratique jusqu'à un certain moment, alors ses créateurs ont décidé de la faire évoluer : c'est la naissance de RIP

version 2 (RIPv2). Cette dernière supporte l'implémentation des masques de sous-réseaux, ce qui veut dire qu'elle peut efficacement router les paquets dans un réseau basé sur l'adressage CIDR. L'échange des tables de routage se fait par *multicast*. Les tables de routage sont transmises aux autres routeurs adjacents à l'adresse 224.0.0.9.

L'autre grande avancée de RIPv2 est qu'on peut désormais sécuriser l'accès à un routeur en utilisant une authentification chiffrée.

##### IV.7.1.3. Côté technique

Ces présentations étant faites, explorons un peu ce protocole d'un point de vue technique.

Nous avons déjà vu que l'unité de métrique du protocole IP était appelée « saut ». Voilà une caractéristique de RIP : pour empêcher les boucles de routage, RIP limite le nombre de sauts par chemin de la source à la destination. Nous avons vu que la limite de sauts était de 15. Notons aussi que RIP utilise le protocole UDP que nous avons étudié dans la couche transport et 520 comme numéro de port.

Le principal rôle des protocoles de routage est l'échange d'informations sur les routes. Que se passe-t-il alors dans un réseau si une route ne fonctionne plus ? Les routeurs continueront à s'échanger des tables de routage. Cela veut donc dire que des informations fausses pourront être transmises à d'autres routeurs. Pour empêcher la diffusion de ces informations inexactes, RIP utilise trois mécanismes distincts : *split horizon*, *route poisoning* et finalement le *holddown*. Voyons en détail cette première technique, *split horizon*.

##### IV.7.1.4. Séparation horizontale

Cette technique, qui est d'ailleurs reprise par d'autres protocoles comme IGRP et EIGRP, empêche un routeur de diffuser des fausses informations en l'interdisant de transmettre sa table de routage par l'interface au travers de laquelle il a reçu la dernière mise à jour de sa table.

C'est peut-être confus, nous allons regarder cela avec un schéma à l'appui pour mieux comprendre.

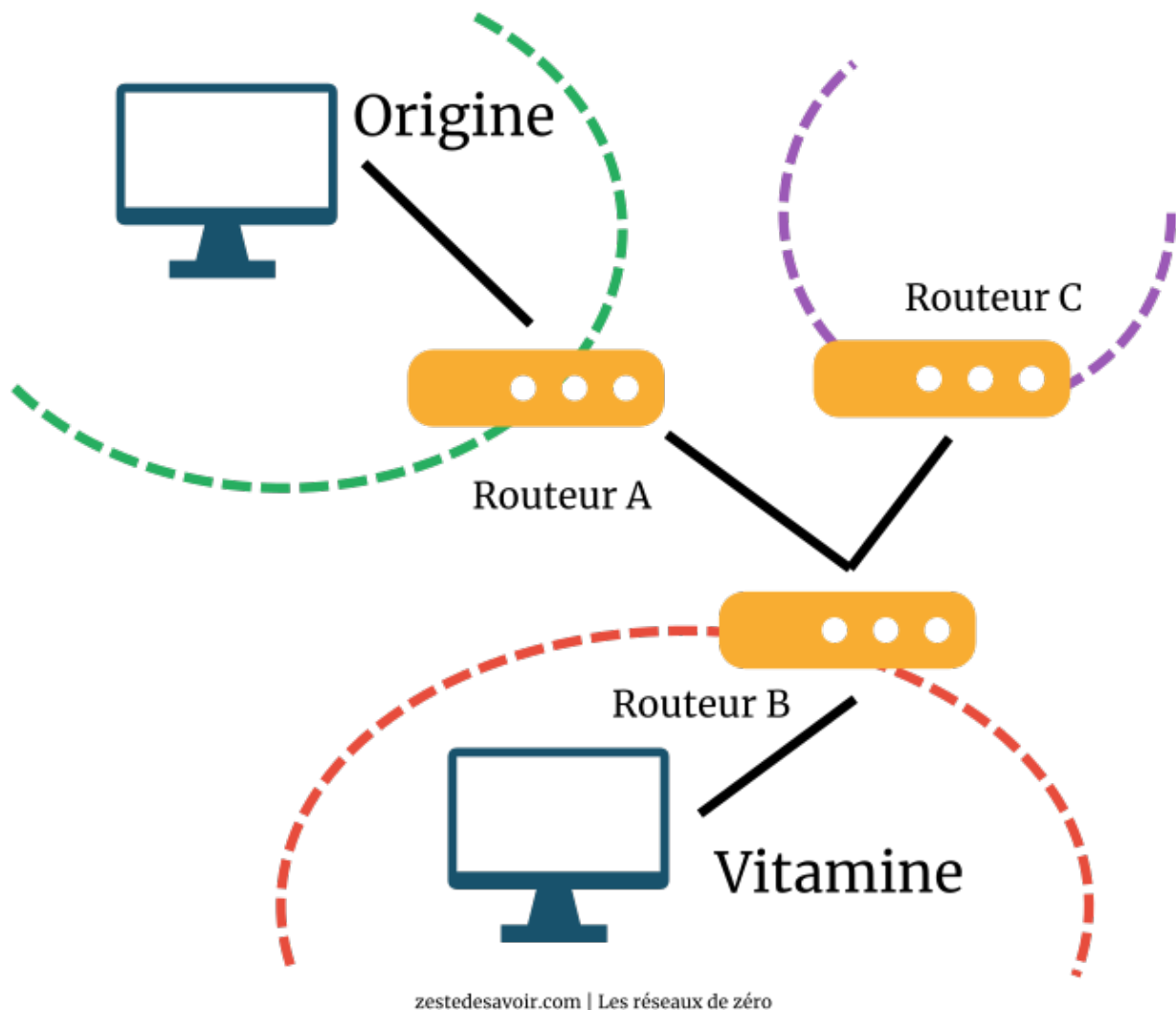


FIGURE IV.7.1. – Trois réseaux, trois routeurs (CC BY)

Dans le schéma ci-dessus, l'échange des tables de routage devrait, en principe, se faire comme suit :

- A envoie sa table à B ;
- B met sa table à jour et la renvoie à A ;
- B envoie sa table à C ;
- C met sa table à jour et la renvoie à B.

Maintenant, que se passerait-il si après la deuxième étape, le routeur A tombait en panne, comme illustré dans le schéma ci-dessous ?

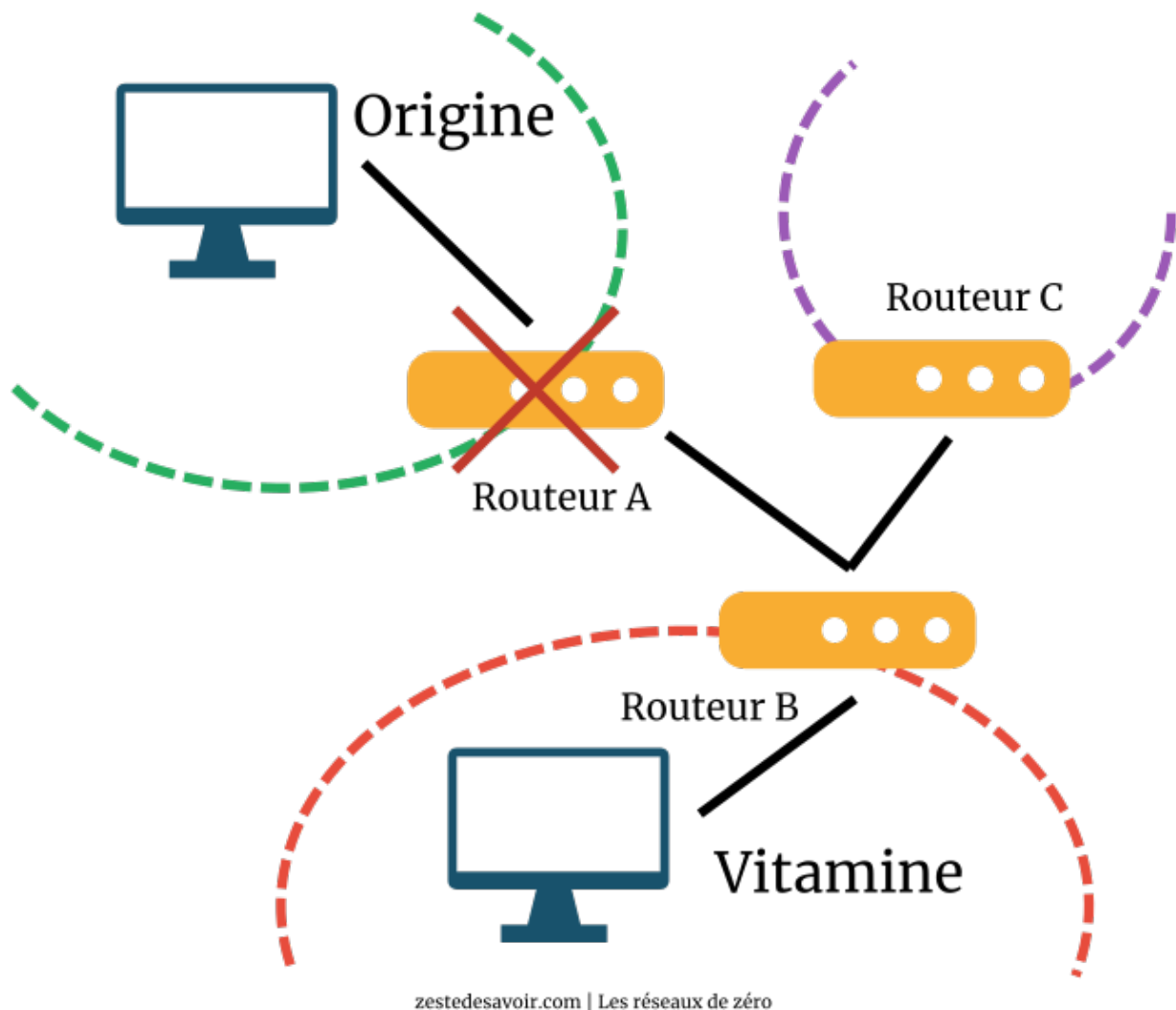


FIGURE IV.7.2. – Le routeur A tombe en panne (CC BY)

En toute logique, C va mettre sa table à jour comme l'indique l'étape 4 et ensuite renvoyer sa table à B ! 🍊

?

Et quel est le problème ?

Eh bien, quand B a envoyé sa table à C, elle contenait la route vers le réseau Origine, auquel A est directement relié. Mais, si le routeur A est en panne, cela veut dire que la route vers Origine n'existera plus. C va alors transmettre de fausses informations à B, car la route menant à A n'existe plus !

?

Malheur ! Comment éviter un tel drame ? 🍊

C'est justement le *split horizon* qui fera ça. Par ce mécanisme, RIP va interdire à C de renvoyer sa table de routage à B, puisque, de toutes les façons, aucun paquet ne sera transmis au réseau Vitamine via A. C et B sont directement reliés, donc ça ne servirait à rien de passer par A pour transmettre un paquet au réseau Vitamine. 🍊

#### IV. Dans les basses couches du modèle OSI

Avant de passer à la conclusion de cette section, un mot sur les autres techniques pour éviter de pourrir les tables de routage. Le *route poisoning* (empoisonnement de route) consiste à combattre le feu par le feu. 🐱. Un routeur qui détecte la perte d'une connexion à un réseau adjacent, comme A à Origine ou B à Vitamine, peut immédiatement mettre à jour la route vers ce réseau en indiquant un nombre de sauts de 16. Comme cette valeur n'est pas permise par RIP, les routeurs considéreront la route comme invalide et ne la prendront plus en compte. Le *holddown* consiste à être prudent : quand une information de route injoignable est reçue, un compte à rebours se déclenche, durant lequel sera ignoré tout message indiquant que la route est joignable. Cela permet de ne pas prendre en compte une information qui parviendrait avec un délai et qui invaliderait à tort le message d'indisponibilité.

##### IV.7.1.5. Comparaison finale

Pour terminer sur RIP, voici un tableau comparant les deux versions.

Propriété	RIPv1	RIPv2
Famille de protocole	IGP	IGP
Intervalle de mise à jour	30 secondes	30 secondes
Système d'adressage	Par classes	Sans classe
Algorithme de base	Bellman-Ford	Bellman-Ford
Adresse de mise à jour	Broadcast sur 255.255.255.255	Multicast sur 224.0.0.9
Protocole et port	UDP 520	UDP 520
Unité de métrique	Saut (jusqu'à 15)	Saut (jusqu'à 15)
Élément de mise à jour	Table entière	Table entière

Le tableau ci-dessus n'est pas complet, nous n'avons pas représenté toutes les caractéristiques de RIP. Par exemple, nous n'avons pas spécifié la valeur par défaut de la distance administrative, qui constitue également une métrique. Toutefois, cette métrique n'existe que sur les routeurs Cisco, nous n'avons pas trouvé nécessairement opportun d'en parler.

##### IV.7.2. Vecteur de distances

Nous avons dit que RIP, comme la majorité des protocoles de routage à vecteur de distances, sont basés sur l'algorithme de Bellman-Ford. C'est ce principe algorithmique que nous allons maintenant explorer. 🍊

Dans un réseau basé sur un protocole de routage à vecteur de distance (VD, ou aussi *distance vector*, DV), le chemin que doit emprunter un paquet est déterminé par deux critères :

- La distance : dans un algorithme VD, la distance entre deux réseaux est la valeur de l'unité de métrique qui les sépare. Cette unité, pour les protocoles VD, correspond généralement au nombre de sauts. Ainsi, la distance entre un réseau A et un réseau B est déterminée par le nombre de routeurs qui les sépare.
- La direction : ce critère est déterminé par deux facteurs.

#### IV. Dans les basses couches du modèle OSI

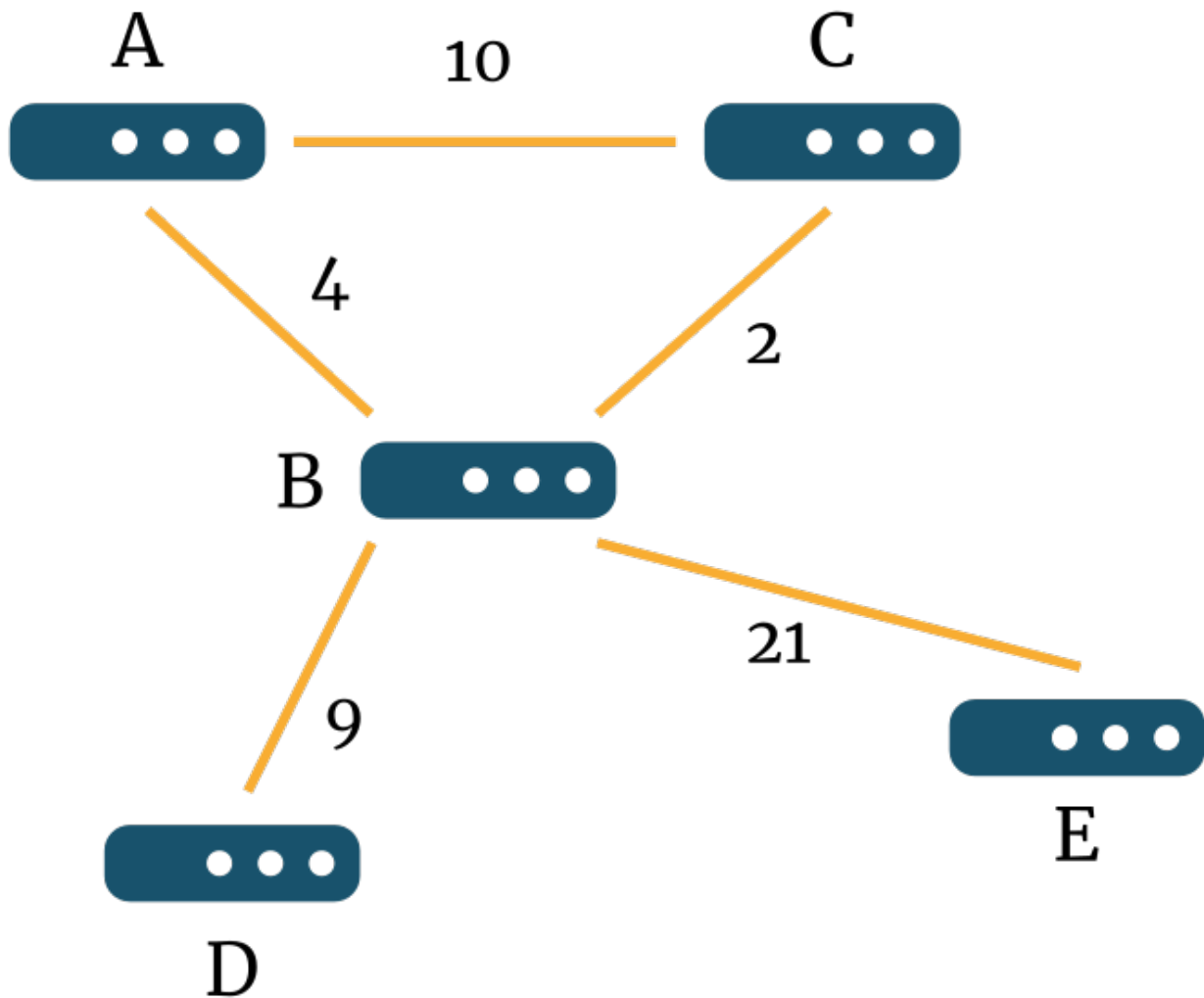
- Le *next hop* : quand nous avons vu ce qu'était une table de routage, nous avons dit que le *next hop* ou prochain saut était l'adresse IP du routeur auquel les paquets devaient être routés.
- L'interface de sortie : il s'agit de l'interface par laquelle sortiront les paquets routés vers le *next hop* de la table de routage.

Dans un réseau pareil, les routeurs sont représentés sous forme de vecteurs de direction et de distance. Le contenu de leurs tables de routage résulte du calcul de la direction et de la distance entre réseaux. Les routeurs ne connaissent pas tous les chemins possibles, car lors des échanges de table de routage, seuls des échanges directs s'effectuent. En d'autres termes, les routeurs n'échangent leur table qu'avec les routeurs auxquels ils sont directement reliés. Ils n'ont pas de vision globale du réseau. Cette philosophie de routage est également appelée « **routage par rumeur** ». Par « rumeur », la pensée exprimée est qu'on ne sait pas si une information est vraie ou fausse. Les routeurs dans un réseau basé sur un protocole de routage VD s'appuient sur les informations reçues des routeurs voisins, auxquels ils sont directement reliés, pour assurer le routage, sans vérifier leur véracité.

##### IV.7.2.1. Le principe d'un algorithme VD

Nous allons voir comment fonctionne un algorithme de routage à vecteur de distances. C'est une notion un peu difficile à appréhender, nous allons donc l'illustrer au mieux et le plus précisément possible (comme toujours 🍊).

Nous allons considérer le réseau suivant, constitué des routeurs A, B, C, D, et E. Ces 5 routeurs sont reliés entre eux comme le montre le schéma ci-dessous. Les traits orange représentent les connexions entre les routeurs, mais attention ! Ils ne sont pas directement reliés entre eux.



zestedesavoir.com | Les réseaux de zéro

FIGURE IV.7.3. – Interconnexion de 5 routeurs avec des sauts entre chaque (CC BY)

Comme l'indique le schéma, il y a des sauts entre chaque routeur. Par exemple, entre les routeurs A et C, il y a 10 sauts, ce qui indique qu'il y a *a priori* 10 routeurs qui les séparent. Nous n'allions quand même pas tous les représenter. 🍎 Nous allons utiliser ce réseau de base pour analyser comment le meilleur chemin est déterminé par cet algorithme VD.

#### IV.7.2.2. Déterminons les chemins actuels

À partir de A	via A	via B	via C	via D	via E
allant vers A					
allant vers B		4			
allant vers C			10		

IV. Dans les basses couches du modèle OSI

allant vers D					
allant vers E					

À partir de B	via A	via B	via C	via D	via E
allant vers A	4				
allant vers B					
allant vers C			2		
allant vers D				9	
allant vers E					21

À partir de C	via A	via B	via C	via D	via E
allant vers A	10				
allant vers B		2			
allant vers C					
allant vers D					
allant vers E					

À partir de D	via A	via B	via C	via D	via E
allant vers A					
allant vers B		9			
allant vers C					

#### IV. Dans les basses couches du modèle OSI

allant vers D					
allant vers E					

À partir de E	via A	via B	via C	via D	via E
allant vers A					
allant vers B		21			
allant vers C					
allant vers D					
allant vers E					

Après initialisation, les routeurs vont « découvrir » l'environnement dans lequel ils se trouvent. Cette « découverte » leur permettra de commencer à remplir leur table de routage. Comme vous le voyez, le routeur A ne connaît que le chemin vers ses routeurs voisins, soit B et C. Pour aller à B, il peut y aller directement et cela ne lui coûte que 4 sauts. Pour aller à C, cela coûte 10 sauts. À ce stade précis, pour le routeur A, C est le meilleur chemin (étant donné que c'est l'unique) pour arriver au réseau C, et B également est le meilleur chemin pour arriver au réseau B.

*i*

Nous n'avons pas représenté les différents réseaux que ces routeurs lient. En pratique, A dispose d'une interface liée au réseau A, B a une interface liée au réseau B, etc.

Le routeur B de son côté fera la même chose. Il va remplir sa table en fonction de ses routeurs voisins. Étant donné que B est le routeur « central », il aura dès le départ plus d'informations que les autres. Il sait qu'il existe un chemin menant à A en passant directement par A, et qu'il coûte 4 sauts. Il existe également un chemin menant à C en passant directement par C qui coûte 2 sauts. Le reste des données du tableau devrait vous paraître évident.

Le routeur C, quant à lui, est relié aux routeurs A et B. Il va donc remplir sa table en fonction de ces liaisons. Il faut 10 sauts pour arriver à A en passant par A directement et 2 sauts pour atteindre B directement. À ce stade également, le vecteur A10 constitue le meilleur chemin pour atteindre A, de même que B2 est le meilleur chemin pour atteindre B. 🍌

*i*

L'appellation « vecteur A10 », par exemple, vient du fait que chaque routeur est considéré comme étant un vecteur de distance. Le vecteur est donné par la direction (le routeur vers lequel on se dirige, A en l'occurrence) et la distance (le nombre de sauts, 10 en l'occurrence). Donc, pour le routeur D par exemple, B9 est le routeur B, éloigné de 9 sauts.



Le routeur D ainsi que le routeur E sont isolés, les pauvres ! 🍊 Ils ne connaîtront qu'une seule route directe chacun et cette route passe obligatoirement par B. D y est relié par une route qui coûte 9 sauts, et il en faut 21 pour qu'un paquet venant du réseau E atteigne le réseau B.

#### IV.7.2.3. Premier échange des VD

À partir de A	via A	via B	via C	via D	via E
allant vers A					
allant vers B		4	12		
allant vers C		6	10		
allant vers D		13			
allant vers E		25			

À partir de B	via A	via B	via C	via D	via E
allant vers A	4		12		
allant vers B					
allant vers C	14		2		
allant vers D				9	
allant vers E					21

À partir de C	via A	via B	via C	via D	via E
allant vers A	10	6			
allant vers B	6	2			

#### IV. Dans les basses couches du modèle OSI

allant vers C					
allant vers D		<b>11</b>			
allant vers E		<b>23</b>			

À partir de D	via A	via B	via C	via D	via E
allant vers A		<b>13</b>			
allant vers B		9			
allant vers C		<b>11</b>			
allant vers D					
allant vers E		<b>30</b>			

À partir de E	via A	via B	via C	via D	via E
allant vers A		<b>25</b>			
allant vers B		21			
allant vers C		<b>23</b>			
allant vers D		<b>30</b>			
allant vers E					

Après un laps de temps (30 secondes en ce qui concerne RIP), les routeurs vont s'échanger les informations qu'ils ont reçues lors de la découverte de leur environnement.

B enverra donc sa table de routage aux routeurs auxquels il est connecté, soit A, C, D, et E, et ces derniers pourront mettre leurs tables à jour.

Jetons un coup d'œil à la mise à jour effectuée par A.

A découvre qu'il existe une route vers un réseau D. Mais pour atteindre cette route, il faut passer par B. Étant donné que les routeurs sont représentés comme des vecteurs de distance, comme nous l'avons vu, A pourra alors calculer le nombre de sauts qui le sépare de D. Dans la première étape, il a découvert qu'il pouvait accéder à un réseau B au prix de 4 sauts. Lorsqu'il reçoit la table de routage de B, ce dernier l'informe qu'il est relié à un réseau D avec pour coût 9 sauts.

#### IV. Dans les basses couches du modèle OSI

Ce qui nous donne :

- $A \rightarrow B = 4$
- $B \rightarrow D = 9$
- Donc  $A \rightarrow D = A \rightarrow B + B \rightarrow D$
- $A \rightarrow D = B(4+9) = B13$

Le routeur A devient alors un vecteur B13, ce qui veut dire que pour accéder à D, A doit passer par B et cela coûtera 13 sauts. 🍊 Par la même mise à jour, A découvre également qu'une autre route existe vers un réseau C.

?

Mais lors de la découverte, il le savait déjà, non ? Pourquoi est-ce qu'il considère une nouvelle route vers un réseau qu'il connaît déjà ?

Le but d'un protocole de routage est de connaître tous les chemins possibles pour atteindre une route donnée, pour la redondance et la rapidité. La route découverte par A vers C lors de l'étape 1 valait 10 couts, n'est-ce pas ? Mais B envoie à A sa table de routage et lui dit « Tiens, j'ai découvert un chemin menant à C et cela ne me coûte que 2 sauts ». A va calculer la distance entre lui et C via B, et il se rend compte que ça ne lui coûte que 6 sauts. Il enregistre donc cette nouvelle route dans sa table de routage, et cette dernière deviendra la plus courte pour atteindre C. Au lieu d'aller à C directement, ce qui coûte 10 sauts, il est plus rentable de passer par B, ce qui ne coûtera que 6 sauts. 🍊

!

A ne supprime pas la route initiale menant à C qu'il a découverte dans l'étape 1. Il faut garder toutes les routes possibles menant vers un réseau voisin.

Quant à B, il découvre après avoir reçu la table de routage de C, qu'il existe une autre route vers A, mais en passant par C. Il va donc calculer la distance qui le sépare de A :

- $B \rightarrow C = 2$
- $C \rightarrow A = 10$
- Donc  $B \rightarrow A = B \rightarrow C + C \rightarrow A = C(2+10) = C12$

La nouvelle route conduisant à A coûte 12 sauts en passant par C. B gardera donc la route directe (4 sauts) vers A comme la route la plus rapide, mais ajoutera à sa table le nouvel itinéraire vers A via C.

Le routeur C va découvrir l'existence de 2 nouvelles routes. Il s'agit de D et E, lesquelles routes ne sont accessibles que via B. Sachant, grâce à la table de routage qu'il a reçue de B, que la distance entre B et D est de 9 sauts et celle entre B et E, de 21 sauts, il peut facilement calculer le nombre de sauts le séparant de ces deux réseaux. Cela donne 11 et 23 pour D et E respectivement. Vous devriez avoir compris comment nous avons eu 11 et 23. 🍊

D également découvrira des nouvelles routes, 3 pour être précis. Une route menant à A, une autre menant à C, et une autre menant à E, le tout accessible via B. Il va également calculer la distance qu'il y a entre lui et ces routes et mettre à jour sa table de routage, d'où les nouvelles valeurs que vous pouvez observer dans la table de D. Inutile de détailler les calculs, le principe est le même.

Finalement, E reçoit la table de routage en provenance de C et découvre l'existence des routes A, C et D, accessibles via B. Il fait les calculs pour trouver les valeurs des sauts et met sa table de routage à jour.

Lors de la prochaine mise à jour, il s'agit du même principe : la découverte de tous les chemins

#### IV. Dans les basses couches du modèle OSI

possibles et la mise à jour des tables... en faisant bien attention au *split horizon* : tout chemin qui ferait revenir un routeur vers lui-même ne sera pas pris en compte. Nous n'allons donc pas détailler les 5 tableaux qui suivront. Notez simplement que les tables de routage D et E ne connaissent aucune mise à jour. Ces routeurs n'ont plus rien à apprendre, ce qui est logique car tous les routeurs qu'ils peuvent atteindre passent obligatoirement par B, il n'y a pas d'alternatives.

##### IV.7.2.4. Deuxième échange des VD

À partir de A	via A	via B	via C	via D	via E
allant vers A					
allant vers B		4	12		
allant vers C		6	10		
allant vers D		13	21		
allant vers E		25	33		

À partir de B	via A	via B	via C	via D	via E
allant vers A	4		12		
allant vers B					
allant vers C	14		2		
allant vers D				9	
allant vers E					21

À partir de C	via A	via B	via C	via D	via E
allant vers A	10	6			
allant vers B	14	2			

#### IV. Dans les basses couches du modèle OSI

allant vers C					
allant vers D	<b>23</b>	11			
allant vers E	<b>35</b>	23			

À partir de D	via A	via B	via C	via D	via E
allant vers A		13			
allant vers B		9			
allant vers C		11			
allant vers D					
allant vers E		30			

À partir de E	via A	via B	via C	via D	via E
allant vers A		25			
allant vers B		21			
allant vers C		23			
allant vers D		30			
allant vers E					

Un commentaire rapide de cette étape : A découvre qu'il peut atteindre D et C, non seulement au travers de B, mais également par C. Il fait les calculs et met sa table à jour.

B n'a aucune mise à jour à effectuer, il connaît déjà toutes les routes possibles du réseau.

C découvre qu'il peut atteindre D et E, toujours au travers de B mais en passant par A. En fait, il ne saura même pas que cette nouvelle route passera encore par B. Il reçoit la table de routage de A qui dit « Hé, tu sais quoi ? J'ai découvert un chemin qui mène à D et E ». Il enregistre l'information : « Cool, je peux donc atteindre D et E par A ». C'est tout ce qu'il sait. 🍊 Bien entendu, ces nouveaux itinéraires sont plus coûteux pour C par rapport à un accès direct via B !

#### IV.7.2.5. Suite et fin de l'algorithme

À partir de A	via A	via B	via C	via D	via E
allant vers A					
allant vers B		4	12		
allant vers C		6	10		
allant vers D		13	21		
allant vers E		25	33		

À partir de B	via A	via B	via C	via D	via E
allant vers A	4		12		
allant vers B					
allant vers C	14		2		
allant vers D				9	
allant vers E					21

À partir de C	via A	via B	via C	via D	via E
allant vers A	10	6			
allant vers B	14	2			
allant vers C					
allant vers D	23	11			
allant vers E	35	23			

#### IV. Dans les basses couches du modèle OSI

À partir de D	via A	via B	via C	via D	via E
allant vers A		13			
allant vers B		9			
allant vers C		11			
allant vers D					
allant vers E		30			

À partir de E	via A	via B	via C	via D	via E
allant vers A		25			
allant vers B		21			
allant vers C		23			
allant vers D		30			
allant vers E					

Finalement, nous arrivons à la fin de l'algorithme VD. Il n'y a plus de changement dans la topologie du réseau, ainsi les tables de routage restent les mêmes à ce stade. Nous avons coché les meilleurs chemins pour chaque ligne : il s'agit juste de choisir la route ayant la plus petite métrique.

Alors, cette exploration vous a plu ? 🍊

Sans vouloir vous effrayer, l'algorithme de Dijkstra est encore plus intéressant. Le protocole OSPF est basé sur cet algorithme. Nous continuons avec la même optique : d'abord, on voit le protocole de routage, ensuite, l'algo. 🍊

### IV.7.3. OSPF, le mastodonte

Mastodonte, le mot n'est pas galvaudé. RIP, c'est bien pour de petits réseaux avec peu d'exigences, c'est facile à mettre en place et simple à configurer. OSPF, c'est tout l'inverse. 🍊 On peut passer des heures sur le sujet tellement il est complexe. Dans ce chapitre, nous resterons en surface et nous vous orienterons vers d'autres ressources pour compléter.

Pour commencer, OSPF est un protocole de routage dynamique de type IGP. C'est à peu près le seul point commun avec RIP. Il rentre dans la catégorie des protocoles à **état de liens** (*link*

*state protocol*) et utilise l’[algorithme de Dijkstra](#) [↗](#). On garde ces notions de côté ; pour le moment, on va rester sur l’aspect purement réseau de la bête.

Comme ce protocole est conçu pour de vastes réseaux, il introduit la notion de **zone**, aussi parfois appelée **aire** (*area* en anglais). Chaque zone porte un numéro. L’aire principale porte toujours le numéro 0. Toutes les zones ne sont pas nécessairement reliées entre elles directement, mais une zone OSPF ne peut pas être isolée par des routeurs non OSPF, sinon, elle n’est pas visible du reste du réseau. Et pour cause : les informations sont transmises de proche en proche, directement d’un routeur à l’autre.

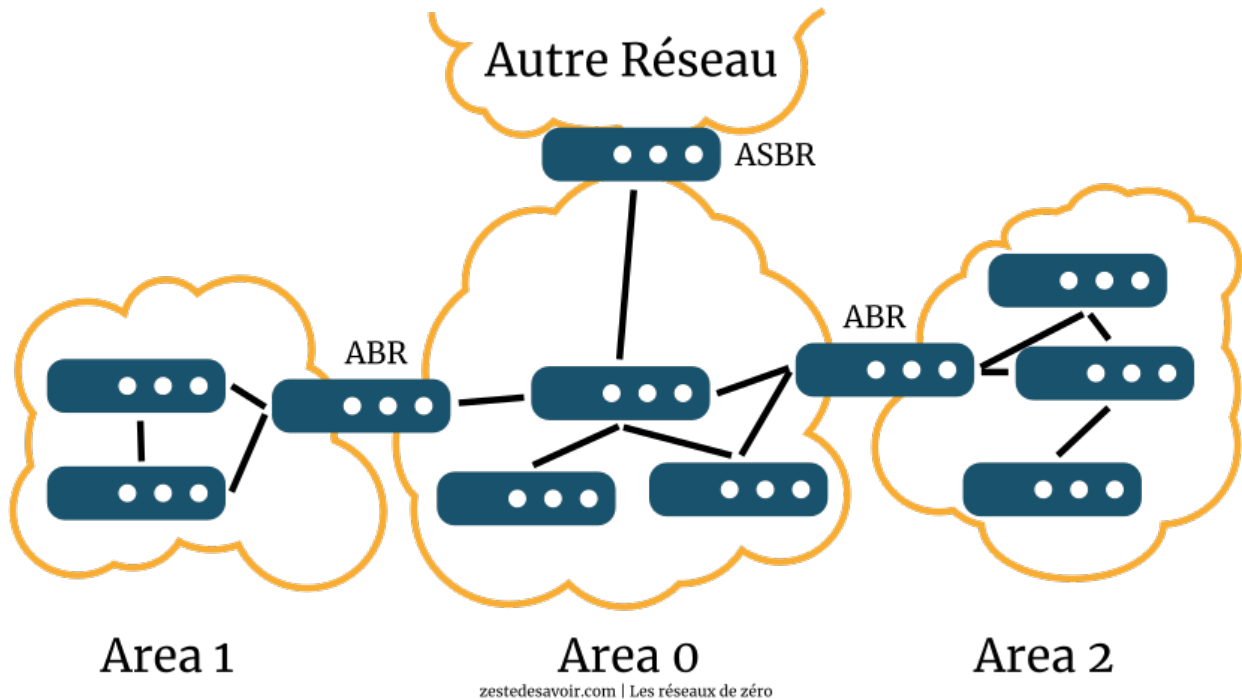


FIGURE IV.7.4. – Illustration d’un réseau OSPF (CC BY)

OSPF a son vocabulaire propre, et franchement, ce n’est pas très grave si vous ne le reprenez pas. L’important, c’est que vous compreniez comment ça marche ! 🍌 Sur l’image ci-dessus, vous pouvez voir des ABR (*Area Border Router*) : ce sont les routeurs qui sont à cheval sur plusieurs aires. On aperçoit également un ASBR (*Autonomous System Boundary Router*) : il s’agit d’un routeur qui est en limite d’un réseau OSPF et qui assure les échanges avec un réseau utilisant un autre protocole de routage, comme RIP ou BGP.

Pour découvrir son environnement, OSPF envoie des paquets *Hello* à ses voisins. Cela permet d’identifier sur quelles interfaces il va avoir à communiquer.



Un routeur A est un **voisin** ou une **adjacence** d’un routeur B si A et B sont directement reliés de manière logique entre eux.

Une fois les voisins découverts, OSPF émet un *Link-State Advertisement* (LSA) en leur direction pour les informer des informations de routage dont il dispose. Les voisins propagent l’information à leurs propres voisins avec un *Link-State Update* (LSU), et ainsi de suite.




Chaque changement dans un routeur génère un LSA, chaque mise à jour dans une table de routage génère un LSU. Contrairement à RIP, aucune information de routage n’est transmise par

#### IV. Dans les basses couches du modèle OSI

OSPF s'il n'y a pas eu de changement. Tout au plus, il envoie des paquets *Hello* régulièrement. Cela permet de signaler aux voisins que le routeur est toujours là, bien que discret. Quand un routeur ne donne plus de nouvelles après un certain temps (de 40 secondes à 2 minutes selon les configurations), il est considéré comme ne faisant plus partie du réseau par OSPF et ne reçoit plus les mises à jour.

Pour attribuer un coût à une liaison, nous avons vu que RIP se base sur le nombre de sauts. OSPF se base sur la bande passante : plus elle est élevée, plus le coût est faible. Une technique utilisée par Cisco consiste à diviser une bande passante de référence par la bande passante réelle. Initialement, la référence était de 100 Mb/s. Vu les débits d'aujourd'hui, cette valeur n'a plus grand intérêt. On peut la configurer pour qu'elle soit, par exemple, de 10 Gb/s, ce qui serait plus pertinent. Ainsi, un lien de 10 Mb/s aura un coût de 1.000 ( $10\text{ G} / 10\text{ M} = 1.000$ ), un lien de 200 Mb/s aura un coût de 50, etc.

Il faut ensuite interpréter toutes ces valeurs reçues pour construire une table de routage. Pour cela, OSPF a recours à l'algorithme de Dijkstra, que nous allons voir dans la section suivante. Avant cela, voici quelques liens pour approfondir votre connaissance d'OSPF. Comme évoqué, ce protocole est un mastodonte, nous n'avons fait que survoler quelques notions fondamentales. C'est *Les réseaux de zéro* ici, pas *Les réseaux vers l'infini et au-delà* ! 🍊


- [Introduction à OSPF](#) , François Goffinet. Un bon article pour introduire de manière plus précise d'autres notions.
- [Open Shortest Path First](#) , Wikipédia. Moins bien structuré mais plus détaillé dans les notions évoquées.
- [Introduction à OSPF](#) , Réussir son CCNA. Plus formel et pratique, assurez-vous d'avoir bien compris le premier lien avant d'appréhender celui-ci.

Enfin, nous pouvons comparer OSPF et RIP avec le tableau suivant.

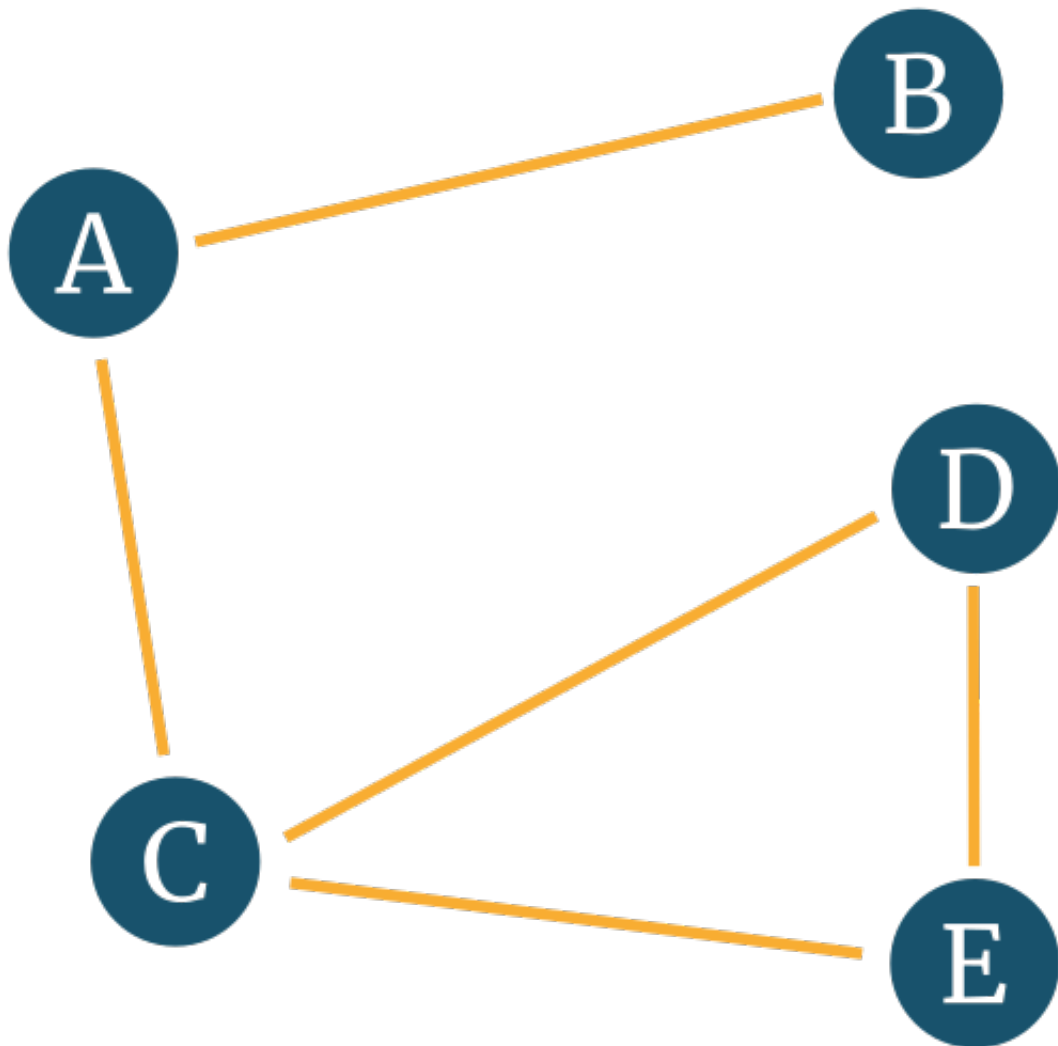
Propriété	RIPv1	RIPv2
Famille de protocole	IGP	IGP
Intervalle de mise à jour	30 secondes	30 secondes
Système d'adressage	Par classes	Sans classe
Algorithme de base	Bellman-Ford	Bellman-Ford
Adresse de mise à jour	Broadcast sur 255.255.255.255	Multicast sur 224.0.0.9
Protocole et port	UDP 520	UDP 520
Unité de métrique	Saut (jusqu'à 15)	Saut (jusqu'à 15)
Élément de mise à jour	Table entière	Table entière

### IV.7.4. Protocoles à états de lien

#### IV.7.4.1. Notions de graphe

Avant tout, nous devons introduire une notion : celle des **graphes**. Ce concept mathématique a bien des applications ; aussi, nous n'introduirons que ce qui est nécessaire à une transposition au réseau. Si vous souhaitez aller plus loin, [vous pouvez vous orienter vers ce tuto](#) .

Un graphe, c'est grossièrement des points reliés par des liens. En mathématiques, on parle de **sommets** ou **nœuds** reliés par des **arêtes**.



zestedesavoir.com | Les réseaux de zéro

FIGURE IV.7.5. – Un graphe tout simple (CC BY)

Ce qui est intéressant, c'est que ce genre de schéma peut facilement être transposé au réseau. Remplaçons les nœuds par des routeurs et les liens par des câbles.

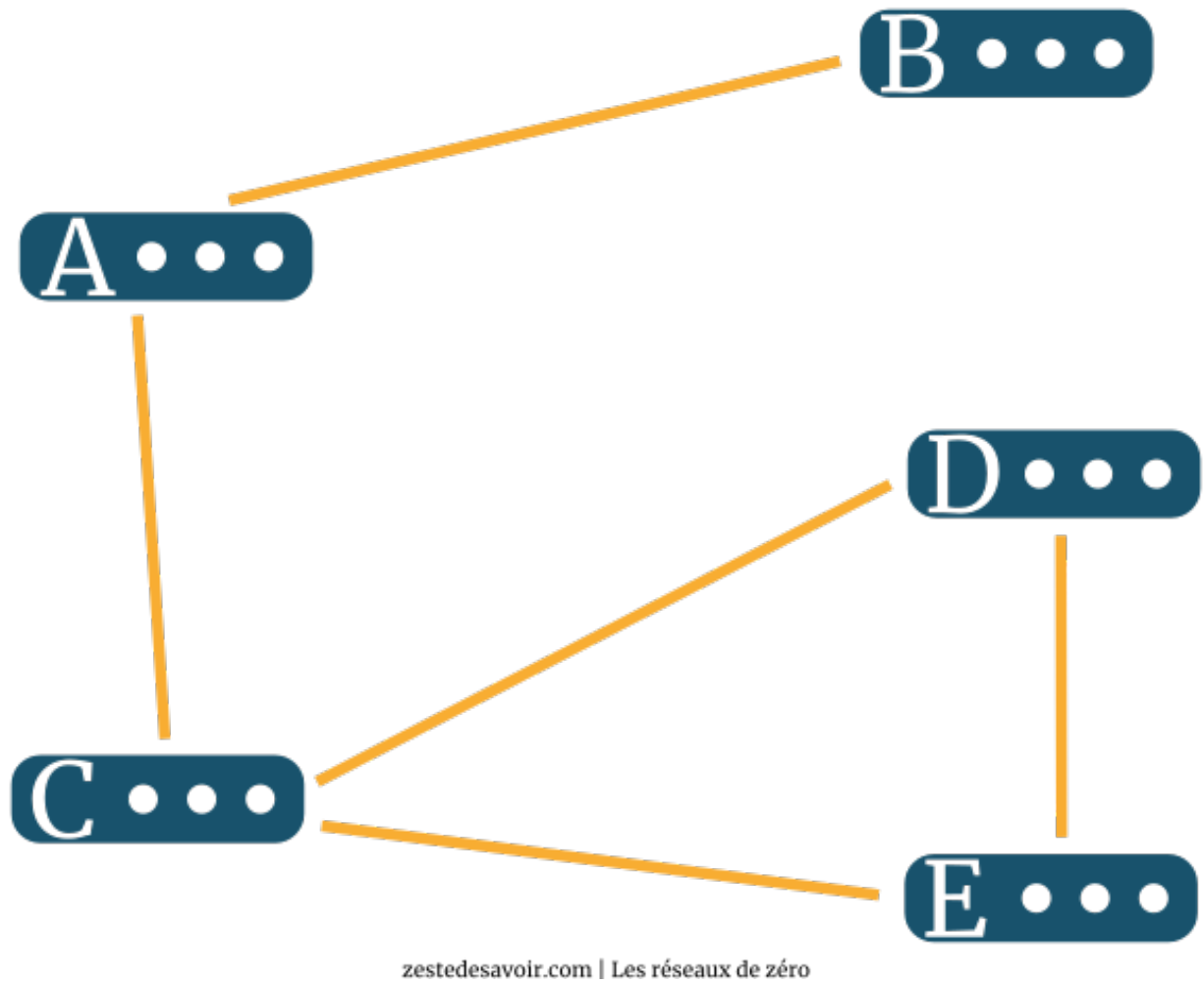


FIGURE IV.7.6. – Un réseau ayant la même structure que le graphe précédent (CC BY)

On peut aussi attribuer des valeurs à ces liens. On parle alors de **poids**. Pondérons le graphe précédent pour l'exemple.

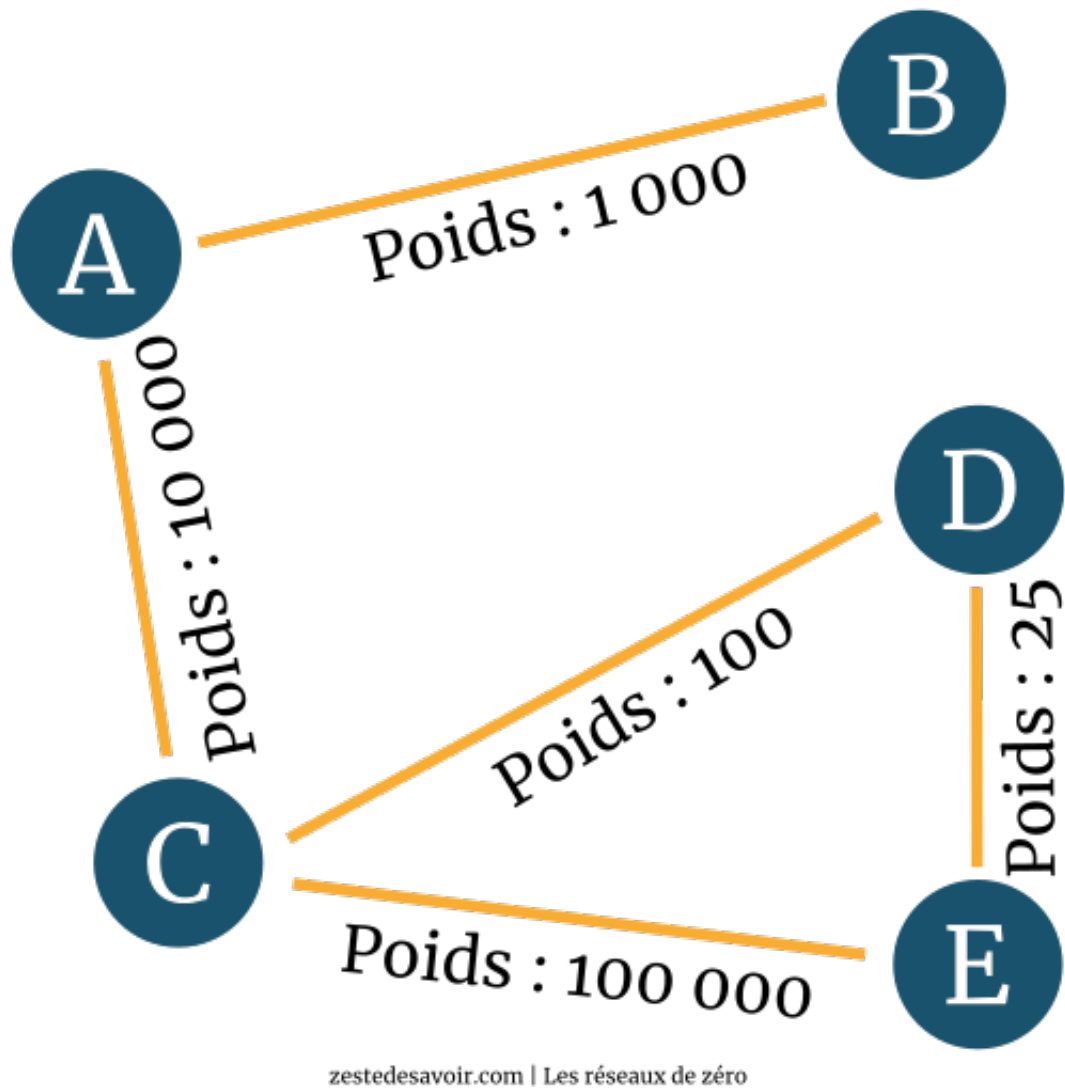


FIGURE IV.7.7. – Un graphe pondéré (avec des poids) (CC BY)

Nous pouvons faire pareil avec le schéma réseau, en y indiquant la bande passante des liens. Mieux : on peut indiquer les poids avec le rapport entre une bande passante de référence et la bande passante des liens, comme vu dans la section précédente. Prenons comme référence 100 Gb/s, soit 100.000 Mb/s.

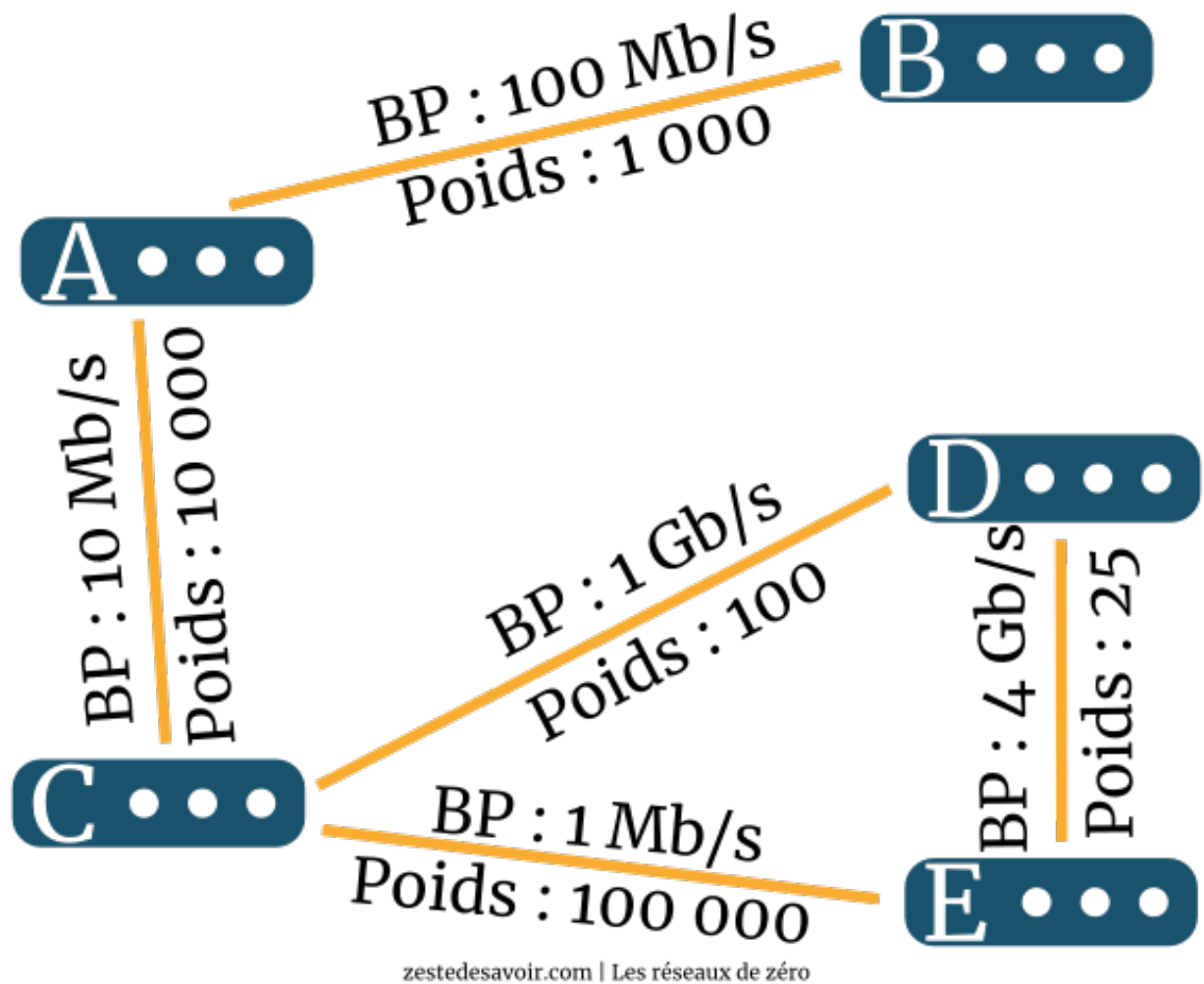


FIGURE IV.7.8. – Un réseau exécutant OSPF (CC BY)

Référence	100.000 Mb/s	100.000 Mb/s	100.000 Mb/s	100.000 Mb/s	100.000 Mb/s
Bande passante	1 Mb/s	10 Mb/s	100 Mb/s	1 Gb/s (1.000 Mb/s)	4 Gb/s
Poids	100.000	10.000	1.000	100	25

*i*

Comme nous transposons directement le concept mathématique à OSPF, nous utiliserons indifféremment les termes de poids et de coût. Dans ce contexte, c'est la même chose.

Ceci étant posé, il est temps de faire connaissance avec votre nouvel ami : Dijkstra.

#### IV.7.4.2. Qui ça ?

Le plus difficile avec Dijkstra, c'est :

1. réussir à l'écrire correctement ;
2. réussir à le prononcer.

#### IV. Dans les basses couches du modèle OSI

Pour le reste, vous avez un cours sous les yeux ! 🍊

Nous allons procéder directement à l'exécution de l'algorithme avec le schéma précédent. Chaque routeur ne calcule que les distances pour lui-même. Prenons le cas du routeur A. Il sait qu'il a, directement connectés, B et C, avec des poids respectifs de 1.000 et 10.000. Sa représentation, pour le moment, se limite à ceci :

via \ Destination	B	C	D	E
B	1.000	$\infty$	$\infty$	$\infty$
C	$\infty$	10.000	$\infty$	$\infty$

TABLE IV.7.25. – Représentation du réseau OSPF pour A avant toute réception d'information

*i*

Quand on ne sait pas le poids qu'aura un chemin, comme ici le chemin de A vers C en passant par B, on le note  $\infty$  (infini). Mathématiquement, cela permet de dire qu'un tel chemin sera toujours plus lourd, et donc moins efficace que n'importe quel autre et qu'il ne faut donc pas l'emprunter.

Faisons le même exercice avec C. Pour le moment, on considère toujours que les routeurs n'ont pas encore transmis d'information par OSPF.

via \ Destination	A	B	D	E
A	10.000	$\infty$	$\infty$	$\infty$
D	$\infty$	$\infty$	100	$\infty$
E	$\infty$	$\infty$	$\infty$	100.000

TABLE IV.7.27. – Représentation du réseau OSPF pour C avant toute réception d'information  
Maintenant, la machine OSPF se met en branle et C envoie sa représentation du réseau à A. Que se passe-t-il dans la tête de A ?

Voyons cela... Pour atteindre A, je m'en fous, c'est moi... Pour atteindre D, le coût depuis C est de 100. Moi, pour atteindre C, j'ai un coût de 10.000. J'additionne les deux, ça me fait un coût total de 10.100 pour atteindre D en passant par C. Mmh, je n'ai pas mieux comme possibilité, j'enregistre cela.

via \ Destination	B	C	D	E
C	$\infty$	10.000	10.100	$\infty$

#### IV. Dans les basses couches du modèle OSI

Il y a autre chose : C peut atteindre E avec un coût de 100.000. Si j'ajoute mon propre coût pour atteindre C, cela donne 110.000. Je n'ai pas de meilleure possibilité, j'enregistre cela.

via \ Destination	B	C	D	E
C	$\infty$	10.000	10.100	110.000

On dirait bien que c'est tout pour le moment.

#### Raisonnement de A

Après l'application de cet algorithme, la représentation OSPF du réseau de A est la suivante :

via \ Destination	B	C	D	E
B	1.000	$\infty$	$\infty$	$\infty$
C	$\infty$	10.000	10.100	110.000

TABLE IV.7.31. – Représentation du réseau OSPF pour A après réception d'informations de C Dans le même temps, D et E vont procéder aux mêmes échanges et aux mêmes calculs. Puis C va aussi envoyer ses informations à D, et ainsi de suite jusqu'à que chaque nœud ait envoyé ses informations, et traité celles qu'il a reçues. Détailler chaque étape est assez long, ce chapitre prendrait des pages et des pages. En revanche, vous, vous pouvez tout à fait faire ce travail et le partager sur le forum pour voir si vous avez tout compris. 🍊 Sachez juste que, quand un routeur calcule un chemin possible qui est moins avantageux qu'un autre chemin qu'il connaît déjà via le même nœud, il va l'ignorer. Par exemple, à un moment, A va recevoir de C un chemin menant à E avec un poids de 10.125. Supposons que l'autre chemin passant par C menant à E avec un poids de 110.000 n'arrive que plus tard. Il serait ignoré, car le premier chemin est plus avantageux.

Vous devriez aboutir au résultat suivant :

via \ Destination	B	C	D	E
B	<b>1.000</b>	$\infty$	$\infty$	$\infty$
C	$\infty$	<b>10.000</b>	<b>10.100</b>	<b>10.125</b>

TABLE IV.7.33. – Représentation du réseau OSPF pour A après computation complète La table de routage de A sera ainsi remplie avec une route pour B qui va directement sur B, ainsi que des routes vers C, D et E qui passent toutes par C. Les coûts associés sont aussi enregistrés dans la table.

via \ Destination	A	C	D	E
A	1.000	11.000	11.100	11.125

TABLE IV.7.35. – Représentation du réseau OSPF pour B après computation complète  
Pour B, c'est encore plus simple : la table de routage est remplie de routes vers A, C, D et E qui passent toutes par A.

via \ Destination	A	B	D	E
A	10.000	11.000	$\infty$	$\infty$
D	$\infty$	$\infty$	100	125
E	$\infty$	$\infty$	100.025	100.000

TABLE IV.7.37. – Représentation du réseau OSPF pour C après computation complète  
Là, c'est un peu plus complexe. Pour atteindre A et B, C n'a d'autre choix que de passer par A, ces routes sont enregistrées dans la table de routage. Par contre, pour aller vers D et E, il y a deux possibilités : passer par D ou par E. Passer par D est moins coûteux pour les deux destinations, on enregistre alors une route vers elles-deux via D. On ne conserve pas les chemins qui passent par E.

via \ Destination	A	B	C	E
C	10.100	11.100	100	100.100
E	110.025	111.025	100.025	25

TABLE IV.7.39. – Représentation du réseau OSPF pour D après computation complète  
Pour D, les routes vers A, B et C passeront par C. Pour atteindre E, il utilisera son lien direct.

via \ Destination	A	B	C	D
C	110.000	111.000	100.000	100.100
D	10.125	11.125	125	25

TABLE IV.7.41. – Représentation du réseau OSPF pour E après computation complète  
Pour E, les paquets à destination de A, B, C et D partiront par D. Le poids très important du lien entre C et E le rend complètement inintéressant.  
Voilà pour Dijkstra ! 🍊 N'oubliez pas que les routes enregistrées dans les tables par OSPF peuvent cohabiter avec d'autres routes et ne seront pas forcément empruntées. En cas de routes

différentes pour une même destination, les protocoles de routage ont un ordre de priorité. Une route OSPF aura l'avantage sur une route RIP, mais elle s'inclinera face à une route statique, c'est-à-dire renseignée manuellement. 🍌

### IV.7.5. BGP : les entrailles d'Internet

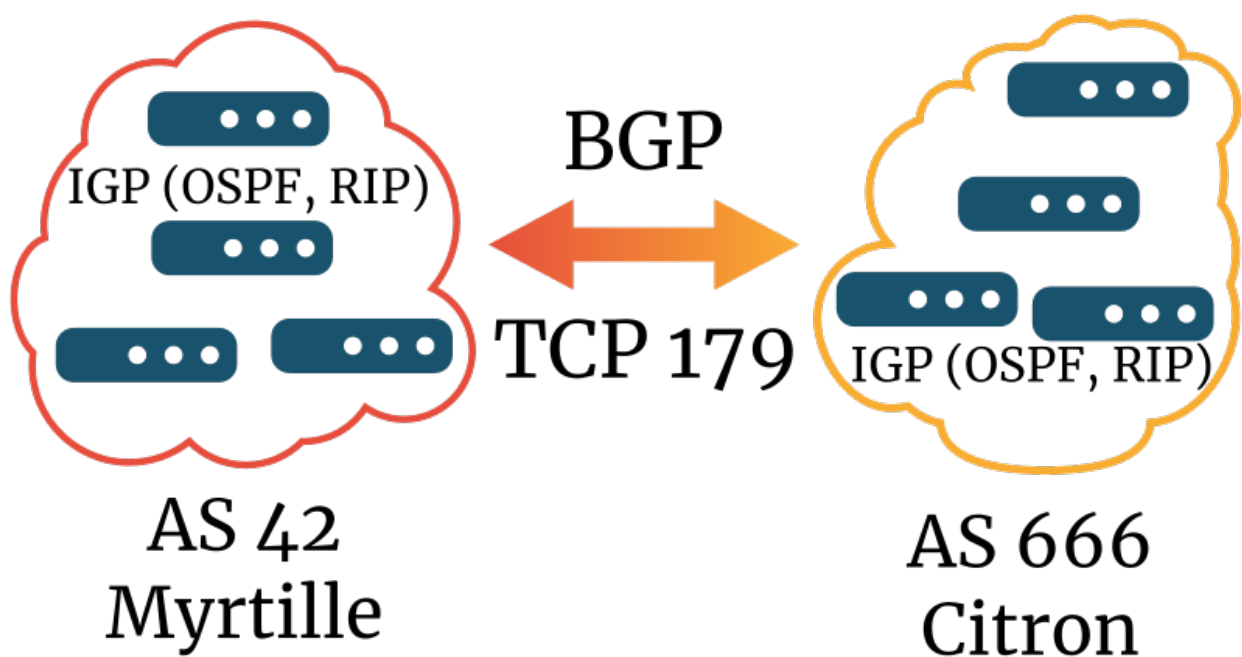
Nous abordons maintenant un protocole particulier. BGP est très différent de ce que nous avons vu jusqu'à présent. Voyons ses caractéristiques dans les grandes lignes.

D'abord, c'est un fondement d'Internet. Son utilité principale est d'assurer l'échange de routes entre fournisseurs d'accès à Internet et autres opérateurs de télécommunications. Ensuite, il fonctionne sur un mode pair-à-pair, en établissant une connexion TCP sur le port 179. Contrairement à RIP ou OSPF qui émettent des messages et découvrent le réseau qui les entoure, BGP doit être spécifiquement configuré sur un routeur pour travailler avec **un seul autre routeur voisin**. On peut avoir plusieurs liaisons BGP (on parle de session) sur un même routeur, mais elles sont indépendantes et doivent être configurées manuellement.

Un routeur qui exécute BGP fait le lien entre un réseau dit interne (les clients d'un **FAI** par exemple) et le réseau d'interconnexion dit externe (Internet). C'est de là que vient le B pour *Border* : ce protocole intervient à la frontière entre ces réseaux. La partie interne est considérée comme un **système autonome**, en anglais *autonomous system* et souvent abrégé AS. Chaque AS dispose d'un numéro qui lui est attribué par une autorité pour éviter les conflits.

i

BGP est donc un protocole de type **EGP**. Il peut aussi être utilisé en interne, au sein d'un AS, par exemple pour des besoins de redondance. Dans ce cas, on parle d' **IBGP**. Dans cette section, nous nous limiterons à l'utilisation externe, dite **EBGP**.



zestedesavoir.com | Les réseaux de zéro

FIGURE IV.7.9. – Illustration du rôle de BGP (CC BY)

#### IV. Dans les basses couches du modèle OSI

Comme on peut s'en douter, sur Internet, le nombre de routes est conséquent. BGP a recours à l'agrégation de routes, [que nous avons étudiée précédemment](#) [↗](#). Cela permet d'alléger un peu les transmissions. Selon le constructeur Cisco, en 2015, une table BGP complète sur Internet compte plus de 570.000 entrées !

Vu le volume, il ne vaut mieux pas balancer toute sa table de routage à son voisin. Pour éviter la saturation, BGP doit être relativement lent. Chaque routeur BGP envoie à ses voisins les modifications de sa table avec un intervalle minimum de 30 secondes par voisin et par préfixe, c'est-à-dire par route agrégée. Cette valeur est une recommandation, ce paramètre peut être modifié. Si une route devient inaccessible, l'annonce est immédiate. Toute route reçue par BGP n'est pas forcément intégrée à la table de routage : des comparaisons sont effectuées avec les entrées existantes pour déterminer si une modification est pertinente.

Comme OSPF, s'il n'a rien à dire, BGP envoie des messages régulièrement à ses voisins pour dire qu'il est en vie. Si un pair n'émet rien durant 90 secondes (valeur par défaut), la session est considérée close par son voisin. 90 secondes, c'est long, mais une fréquence plus élevée implique mécaniquement une augmentation du trafic et donc augmente le risque de congestion. En revanche, en cas de coupure brute, comme une clôture de session niveau TCP ou un câble débranché, la session BGP prend fin immédiatement.

Récapitulons ce que nous avons vu jusqu'à présent avec un tableau.

Propriété	RIPv1	RIPv2
Famille de protocole	IGP	IGP
Intervalle de mise à jour	30 secondes	30 secondes
Système d'adressage	Par classes	Sans classe
Algorithme de base	Bellman-Ford	Bellman-Ford
Adresse de mise à jour	Broadcast sur 255.255.255.255	Multicast sur 224.0.0.5
Protocole et port	UDP 520	UDP 520
Unité de métrique	Saut (jusqu'à 15)	Saut (jusqu'à 15)
Élément de mise à jour	Table entière	Table entière

## Conclusion

Nous avons terminé notre petite étude de quelques protocoles de routage parmi les plus populaires. Il en existe d'autres, comme IS-IS, similaire à OSPF, ou encore EIGRP. Avant de changer de niveau, il nous reste à disséquer le protocole central de la couche réseau !

## IV.8. Le protocole IP

# Introduction

Nous avons étudié la couche 3 et le routage, mais nous n'avons pas encore étudié le protocole IP dans le détail. Ce sera l'objet de ce dernier chapitre sur la couche réseau. Après ce que vous avez subi précédemment, ça va vous paraître tellement simple ! 🍊

### IV.8.1. L'en-tête IPv4

Commençons par regarder à quoi ressemble un paquet IP, en particulier au niveau en-tête.  
*Spoiler* : ça commence par le numéro de version.

?

Ah oui, on avait vu qu'il y a IPv4 et IPv6 ! Mais c'est pas juste une histoire d'adresses ?

Hm, c'est *un peu* plus complexe que ça. 🍊 En premier lieu, regardons l'en-tête d'un paquet IPv4.

Offset								1								2								3								
Offset								1								2								3								
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version		IHL				DSCP				ECN		Longueur totale																				
Identification																Flags		Fragment offset														
Time To Live								Pro- to- cole						Somme de contrôle de l'en- tête																		
Adresse IP source																																

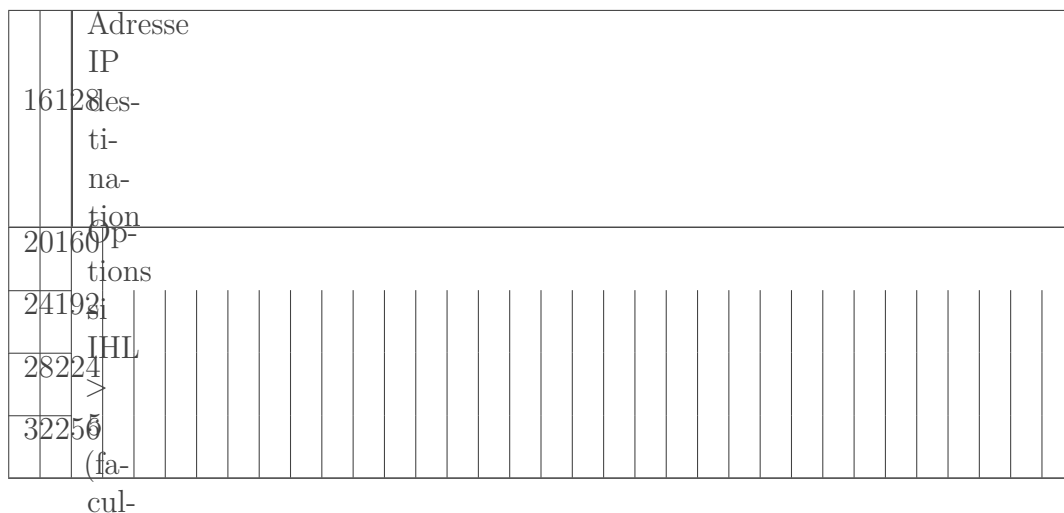


TABLE IV.8.2. – Tableau représentant les champs d'un paquet IP

Comme évoqué, le premier élément qu'on trouve est le numéro de version, sur 4 *bits*. Avec IPv4, ce champ vaut... 4, il n'y a pas de piège.

Vient ensuite l'IHL, pour *Internet Header Length*. Cette information correspond à la longueur de l'en-tête du paquet, et est stockée sur 4 *bits*.

...

...

On peut apercevoir vos mines sceptiques depuis très, très loin. 🍊 Quel est l'intérêt de cette information, pourquoi à cet endroit, comment peut-on indiquer une taille d'en-tête sur 4 *bits*, ce qui donne une valeur maximale de 15, alors qu'il semble bien plus gros, pourquoi certaines clémentines ont des pépins et pas les autres !?

Pour les clémentines, le mystère reste entier. Pour le reste, on peut constater que le protocole IPv4 n'a pas été conçu pour être un modèle de sobriété. IPv4 connaît un système d'options qui viennent s'ajouter au bout de l'en-tête. On ne sait pas, a priori, quelle taille ces options représentent, ni même combien il y en a. Pour délimiter où s'arrête l'en-tête, et donc savoir quand commence le SDU, il faut connaître la longueur du *header*. Le processus de la couche réseau qui reçoit ce paquet s'arrêtera de décapsuler une fois l'en-tête parcouru grâce à cette information.

Maintenant, passons au problème de taille. 4 *bits* pour un en-tête qui fait minimum 20 octets, c'est peu. L'astuce, c'est que cette valeur correspond au nombre de blocs de 4 octets. Autrement dit, il faut multiplier cette valeur par 4 pour savoir le nombre d'octets total de l'en-tête. Et si le *header* ne fait pas un nombre d'octets multiple de 4 à cause d'options, on rajoutera des zéros au bout pour que ça fasse quand même un multiple de 4. Oui, Internet a été conçu en partie par des gens tordus. 🍊

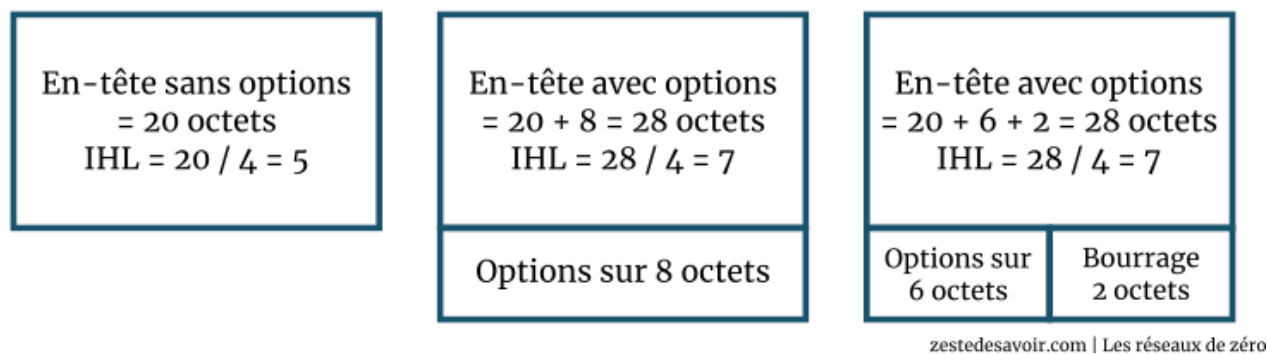


FIGURE IV.8.1. – Représentation de la place définie par l'IHL (CC BY)

Vient ensuite le DSCP, pour *Differentiated Services Code Point*. Il s'agit d'un code, sur 6 octets, qui indique le type de service contenu dans le paquet. Il est utilisé à des fins de qualité de service. Pour bien comprendre de quoi on parle, explicitons ces notions.

Différentes applications n'ont pas nécessairement les mêmes besoins en termes de qualité de réseau pour fonctionner. Nous avons vu, quand nous parlions des protocoles de transport, que la VoIP a besoin que les données soient transmises avec régularité et avec peu de délai. Par contre, des flux de contrôle du réseau peuvent nécessiter d'être transmis le plus rapidement possible mais sans contrainte de gigue. À l'inverse, certaines transmissions comme des e-mails peuvent se permettre d'arriver plus tard, ce mode de communication n'étant pas prévu pour être instantané. On a là divers degrés de **priorité**. C'est tout l'objet de la problématique de la qualité de service, abrégée QoS : comment prioriser certains flux par rapport à d'autres quand il y a des besoins particuliers ? Le champ DSCP est une solution : les routeurs le lisent et, s'ils ne peuvent pas traiter tous les paquets en même temps, vont prioriser en fonction du type de service. Vous vous souvenez des histoires de cousins qui s'envoient des lettres ? La Poste correspondait à la couche réseau, dans notre analogie. Ici, le champ DSCP, c'est un peu comme le type de timbre utilisé : il peut être prioritaire, économique, etc.

Mais, parce qu'il y a un mais, on ne peut pas faire n'importe quoi. Vous n'imaginez quand même pas qu'on peut mettre un *tag* de priorité à nos paquets qu'on envoie sur Internet ? 🍊 Techniquement, on peut, mais notre opérateur aura vite fait de le remettre à zéro. Ce système ne fonctionne que sur un réseau DiffServ, c'est-à-dire qui opère une différenciation entre les services. Si un routeur, à un endroit, a décidé<sup>1</sup> de ne pas faire le DiffServ, au mieux tous les paquets sont traités de manière égale à cet endroit, au pire le champ est remis à zéro. Le DSCP est donc réservé aux réseaux d'entreprise qui ont besoin d'assurer une qualité de service à certaines applications.

Le champ suivant est l'ECN. Nous l'avons déjà étudié avec le [contrôle de congestion](#) . Il n'y a pas grand-chose à ajouter, sinon qu'il est codé sur 2 *bits* et peut donc prendre 4 valeurs différentes. 00 signifie que l'ECN n'est pas supporté, 01 et 10 indiquent tous deux que l'ECN est supporté, et 11 alerte le destinataire d'une congestion sur le réseau, comme expliqué dans le chapitre précédemment cité.

La valeur qui vient ensuite correspond à la longueur totale du paquet, en-tête et données inclus. Pas de bizarreries ici, elle correspond bien au nombre d'octets total et s'exprime sur 2 octets (16 *bits*).

Vous êtes toujours là ? Félicitations, vous savez maintenant interpréter les 4 premiers octets d'un paquet IP ! 🍊 Les 4 octets suivants concernent la fragmentation, nous y consacrerons une section entière. Passons tout de suite à la troisième ligne du tableau représentant le *header*.

1. Ou plutôt, a été configuré pour.

Nous rencontrons alors le TTL. C'est la durée de vie du paquet. Il ne s'agit pas d'un nombre de secondes ou de minutes, mais d'un nombre de sauts. Nous avons vu dans les chapitres sur le routage que lorsqu'un paquet passe un routeur dans le but d'atteindre sa destination, cela correspond à un saut. À chaque saut, le routeur décrémente ce TTL, c'est-à-dire retire 1 à sa valeur. Quand elle atteint zéro... Boum ! Le paquet est jeté comme un malpropre.



Quoi ? Mais c'est dégueulasse ! Ça veut dire que les paquets peuvent disparaître sans raison comme ça, dans l'indifférence générale ? 🍊

Sans raison, certainement pas. 🍊 Ce système existe afin d'éviter qu'un paquet ne se retrouve pris au piège dans une boucle. Si, pour une raison quelconque comme une erreur de configuration, des routeurs sont amenés à se renvoyer indéfiniment un paquet, le pauvre est condamné à tourner tel un satellite en orbite. En étant terre-à-terre, le problème ici est que les routeurs vont travailler et finir par saturer pour rien. Le système de TTL permet d'éviter ces conséquences : au bout d'un certain nombre de sauts, le paquet est jeté. Cette valeur est stockée sur 1 octet, elle peut donc aller jusqu'à 255. Les valeurs les plus courantes sont 64, 128 et 255.

Toutefois, le paquet n'est pas juste effacé. Le routeur qui élimine un paquet envoie un message à l'expéditeur (l'adresse IP source) pour le prévenir que la destination est injoignable. Pour ce faire, il utilise le protocole ICMP, que nous étudierons en fin de chapitre.

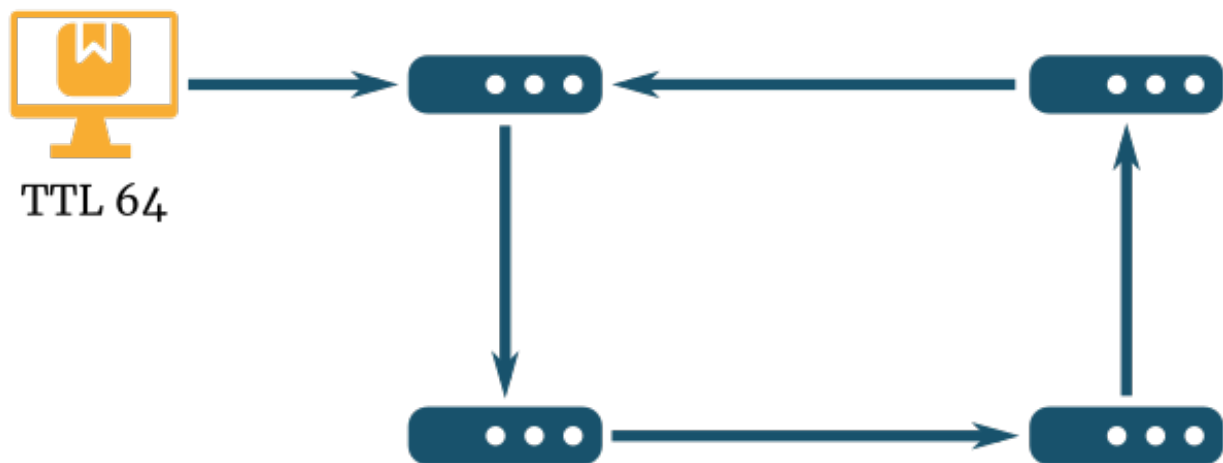


FIGURE IV.8.2. – Boucle de routage (CC BY)

Si l'animation ci-dessus ne s'affiche pas ou ne se lance pas, [cliquez ici](#) .

Le champ suivant porte bien son nom : « protocole ». Il précise quel est le protocole utilisé dans les données utiles. Cela peut être TCP, UDP, OSPF ou bien d'autres . Cette valeur, sur un octet, permet au destinataire de savoir à quel processus transmettre le SDU.

IV. Dans les basses couches du modèle OSI

Nous en arrivons à la somme de contrôle de l'en-tête, en anglais « *header checksum* ». Il s'agit de la somme de contrôle de l'en-tête du paquet. Oui, juste de l'en-tête. IP est un peu fainéant et considère que ce n'est pas à lui de faire la *checksum* de son SDU. Pour ne pas vous noyer, le détail de la méthode utilisée est [dans l'annexe sur les sommes de contrôle](#) ☞ .

*i*

Comme le TTL varie à chaque saut, et qu'il fait partie de l'en-tête, cette somme de contrôle est forcément modifiée par chaque routeur.

On approche de la fin de l'en-tête IPv4. Passons sur les adresses source et destination, vous en avez assez soupé dans la partie 3 du cours.  
Enfin, nous arrivons aux options. Et nous ne nous attarderons pas dessus : soit elles ne servent à rien, soit elles sont expérimentales, soit elles ne sont pas supportées. 🍊 Pour la culture, nous vous renvoyons vers [cette fiche de FrameIP](#) ☞ pour que vous vous fassiez une idée du sujet.  
Un sacré bazar, cet en-tête IPv4. Rassurez-vous, à ce niveau, IPv6 est beaucoup mieux fichu !

IV.8.2. L'en-tête IPv6

Offset								1								2								3								
Offset								1								2								3								
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Ver- sion				Classe de tra- fic								Flow la- bel																		
4	32	Pay- load length														Next hea- der								Hop li- mit								
8	64																															
12	96	Adresse																														
16	128	source																														
20	160																															
24	192	Adresse																														
28	224	des- ti- na- tion																														
32	256																															
36	288																															

TABLE IV.8.4. – Tableau représentant les champs d'un paquet IPv6

Et on entame avec une bonne nouvelle : une bonne partie des champs sont les mêmes que pour IPv4 ! « Version », comme son nom l'indique, correspond à la version d'IP : on aura donc toujours 6 ici. « Classe de trafic » est un champ représenté sur 8 *bits* dans le tableau. En réalité,

#### IV. Dans les basses couches du modèle OSI

les 6 premiers correspondent au DSCP, les 2 derniers à l'ECN. « *Hop limit* » est équivalent au TTL. Quant aux adresses source et destination, inutile de s'y attarder... 🍊  
Les nouveautés sont donc « *flow label* », « *payload length* » et « *next header* ». Commençons par ce dernier. Ce champ sur un octet désigne ce qui se trouve après l'en-tête.

?

C'est-à-dire le protocole encapsulé, comme le champ « protocole » en IPv4, en somme !  
Pourquoi compliquer les choses et appeler ça différemment ? 🍊

Hmm, ce n'est pas exactement ça. 🍊 Dans la plupart des cas, on aura effectivement un code représentant le protocole du SDU. Mais il peut aussi faire référence... à une extension d'en-tête. La structure de l'en-tête IPv6 est beaucoup plus claire et moins fouillis qu'avec IPv4. Ici, le *header* peut être complété par des informations additionnelles, comparable aux options IPv4. On parle alors d'en-tête additionnel. L'avantage, c'est que comme ces « options » sont considérées comme des *headers*, elles incluent aussi un champ « next header », et ainsi de suite ! On a donc une chaîne d'en-têtes beaucoup plus faciles à identifier et séparer qu'en IPv4, d'autant plus que leur taille est toujours de 8 octets ou un multiple (16, 24 octets...).

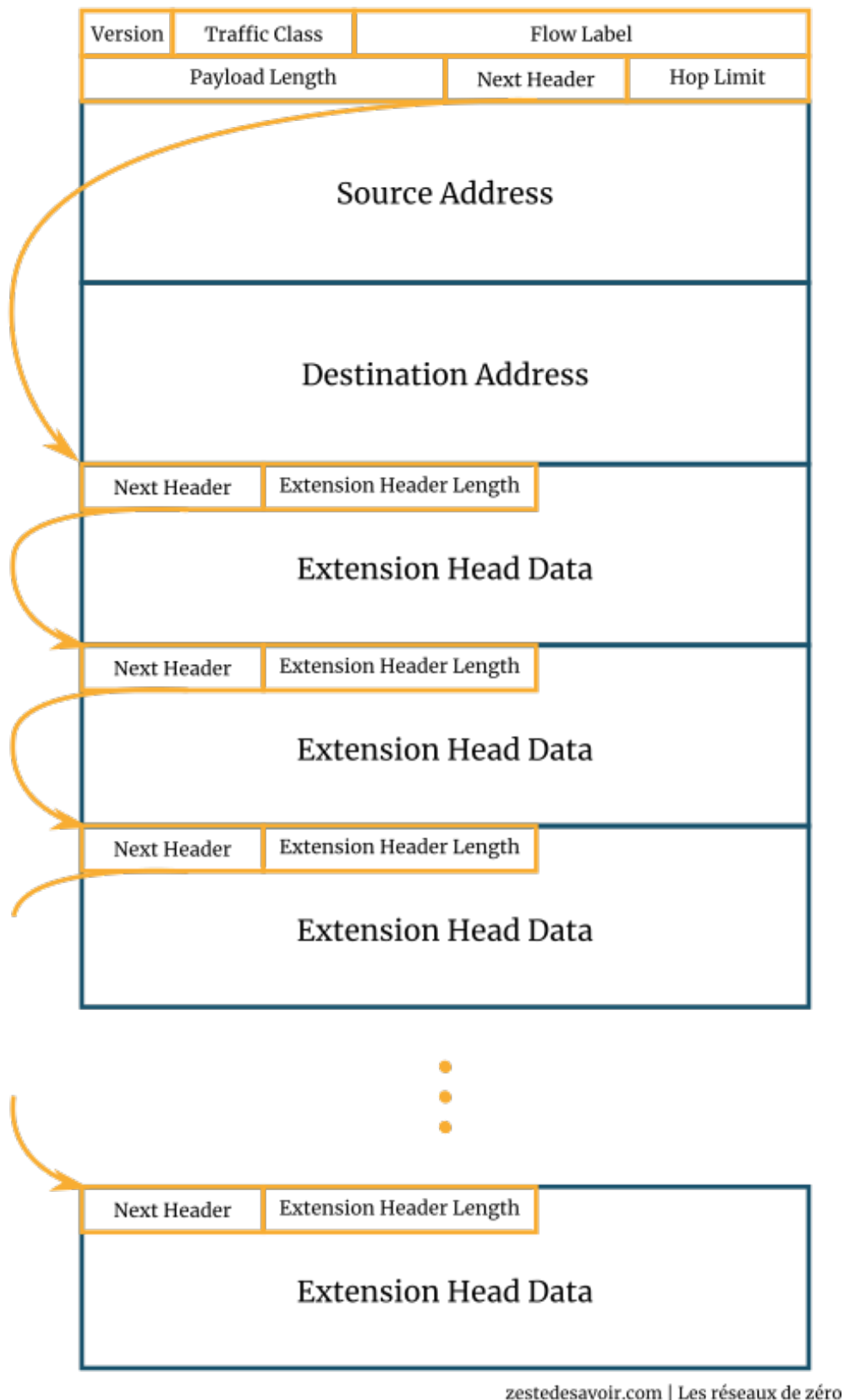


FIGURE IV.8.3. – Structure d'en-tête IPv6 avec extensions (CC BY)

Ce système de chaîne d'en-têtes permet de détacher facilement les éléments du paquet, un peu comme un mille-feuille. Pour le premier *header*, qui contient la plupart des informations, tout ce qui suit est considéré comme *payload* (données utiles). Logique, puisque tout ce qui se trouve

après les adresses IP relève du « prochain en-tête ». La valeur du champ « payload length », sur 2 octets, correspond donc à la taille des données utiles, c'est-à-dire tout ce qui vient après l'adresse de destination. Et pas de bizarrerie ici, cette valeur est bien exprimée en octets.

La dernière nouveauté de l'en-tête IPv6 est le « flow label », littéralement « étiquette de flux ». Ce champ sur 20 *bits* peut prendre n'importe quelle valeur. Il sert à identifier, et plus précisément à étiqueter des paquets qui font partie d'un même flux, comme une communication VoIP. L'intérêt est que les routeurs qui font transiter ce flux peuvent, selon leur configuration, appliquer un traitement particulier. Par exemple, on peut imaginer que des routeurs vont faire en sorte que les paquets d'un flux audio vont arriver dans le même ordre qu'ils ont été émis.

Ce système peut fonctionner dans un réseau d'entreprise, mais nécessite d'appliquer une configuration particulière sur tous les routeurs. Il n'est pas utilisable sur Internet, car on se retrouve avec une problématique comparable au DiffServ en IPv4 : en mettant n'importe quoi comme valeur, on pourrait altérer la qualité de nos communications ou celle d'autres utilisateurs du réseau.

Pour des explications plus détaillées et plus poussées sur le *flow label*, vous pouvez vous référer à la [RFC 6437](#) qui fait autorité à ce sujet.

Nous venons de voir de nombreuses propriétés des paquets IP. Il en est une qui est plus complexe et que nous avons laissée de côté pour le moment : la fragmentation.

### IV.8.3. La fragmentation

Lors de notre passage en revue de l'en-tête IPv4, nous avons éludé la deuxième ligne du tableau, qui concerne la fragmentation. Vous vous demandez certainement, mais qu'est-ce que c'est ?

?

Oui, qu'est-ce que c'est ?

Selon le modèle OSI, les paquets IP sont encapsulés dans un protocole de niveau 2 (liaison de données) avant d'être physiquement envoyés. En fonction du matériel utilisé (câble Ethernet, satellite, ...), le volume de données envoyées en une fois peut être limité. IP doit prendre en considération cela et adapter la taille de ses paquets. C'est pourquoi il va parfois découper un paquet d'origine en plusieurs morceaux plus petits. C'est cela, la fragmentation.

Entre un hôte source et sa destination, les réseaux traversés peuvent être divers. L'émetteur ne sait pas forcément que son paquet va coïncider à un moment. Ainsi, la fragmentation peut avoir lieu pendant le transit. Dans cette éventualité, l'émetteur donne un identifiant à chaque paquet. Ce numéro est codé sur deux octets : c'est le champ **identification**.

Dans l'en-tête, on trouve ensuite 3 *bits* regroupés sous le nom *flags* (drapeaux). Le premier d'entre eux est réservé, il ne sert à rien pour le moment et vaut toujours 0. Le deuxième est le flag **DF** (*Don't Fragment*, ne pas fragmenter). S'il est allumé, les routeurs **ne doivent pas fragmenter le paquet**. S'il est trop gros et ne peut pas passer à un endroit, le routeur concerné le jette et informe l'émetteur que son paquet n'a pas pu être transmis car non fragmentable. Cette information se fait par le protocole ICMP, que nous étudierons dans la section suivante.

*A contrario*, le troisième drapeau est **MF** : *More Fragment* (davantage de fragment). S'il est allumé, cela signifie que le paquet d'origine est fragmenté et que celui-ci n'est pas le dernier.

?

Comment ça ? Qu'est-ce que ça change que ce soit le dernier ?

#### IV. Dans les basses couches du modèle OSI

Avec le protocole IP, il n'y a pas de notion de connexion ou d'acquittement comme avec TCP. Les paquets peuvent arriver dans le désordre voire ne pas arriver du tout. Un *flag* MF allumé permet d'indiquer que, pour un paquet identifié X, il faut s'attendre à davantage de fragments. Pour ce même paquet X, le drapeau MF éteint signifie que c'est le dernier de la série et qu'il ne faut pas en attendre d'autre.

Pour déterminer si le paquet complet peut être reconstitué, IP se base sur le champ *fragment offset* (position du fragment), sur 13 *bits*. Il indique simplement la position du fragment dans le paquet d'origine, sa valeur correspond au nombre d'octets avant lui.

Faisons tout de suite un exemple pour clarifier ces dernières notions quelque peu nébuleuses. Le paquet IP d'origine a pour numéro d'identification **4284**. Il contient uniquement, en plus de son en-tête, les données suivantes : **ABCDEFGHIJKLMN****OPQRST****UVWXYZ**. Cela correspond à 26 octets. Considérons que ce paquet est fragmenté en blocs de 10 octets maximum. Après fragmentation, on aura les 3 paquets suivants :

Identifica- tion	DF	MF	Fragment offset	Données
4284	0	1	0	ABCDEF- GHIJ
4284	0	1	10	KLMNOP- QRST
4284	0	0	20	UVWXYZ

Seuls les champs qui nous intéressent sont représentés dans ce tableau.

Le premier fragment a le *flag* MF levé, car il y en a deux autres à venir derrière. L'*offset* est zéro, car c'est le premier fragment, il n'y en a pas à placer avant lors de la reconstitution.

Le deuxième fragment a aussi le *flag* MF levé, car il y a encore un fragment à venir derrière. L'*offset* est 10, car il y a 10 octets qui doivent être placés avant lors de la reconstitution.

Enfin, le troisième fragment a le *flag* MF baissé, car il n'y a pas davantage de fragment, c'est le dernier de la série. L'*offset* est 20, car il y a 20 octets à placer avant lors de la reconstruction. Il n'y en aura pas plus, le destinataire ne doit pas en attendre d'autre. Bien sûr, si ce 3e fragment n'est pas reçu en dernier, IP va calculer qu'il en manque et les attendre. Mais sitôt le reste reçu, la reconstitution aura lieu.



Pour ne pas vous embrouiller, on a volontairement occulté un détail. En réalité, la valeur du champ *fragment offset* ne s'exprime pas en octets, mais en blocs de 8 octets.

Ça, c'est pour IPv4. Vous aurez peut-être remarqué que ces champs ne se retrouvent pas dans l'en-tête IPv6. Comment gère-t-on alors la fragmentation en IPv6 ?

Eh bien, en réalité, les routeurs IPv6 ne gèrent pas la fragmentation. En IPv6, la responsabilité de la fragmentation incombe forcément à l'émetteur. En cas de paquet trop lourd pendant le routage, le routeur qui pose problème va simplement informer l'expéditeur que son paquet est trop volumineux, puis le jeter. Pour ce faire, il utilise le protocole ICMP. C'est comme quand il reçoit un paquet IPv4 avec le drapeau *Don't Fragment*. Cette technique s'appelle *Path Maximum Transmission Unit Discovery* (PMTUD).

Quand un hôte reçoit ce message ICMP lui disant que son paquet est trop gros, il s'occupe de le découper selon la taille maximale qui lui a été transmise. L'en-tête d'origine reste inchangé, à part le champ *Next header* : on va lui rajouter une option. Dans le cas de la fragmentation, ce

#### IV. Dans les basses couches du modèle OSI

champ prend la valeur 44.

L'en-tête de l'option fragmentation est la suivante :

Offset								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Next header						ré-servé								Frag-ment off-set												ré-servé		MF	
4	32	Iden-ti-fi-ca-tion																													

TABLE IV.8.7. – En-tête additionnel de fragmentation IPv6

Son fonctionnement est similaire à IPv4 : l'identifiant du paquet sur 4 octets, l'offset sur 13 bits, le drapeau MF pour *More Fragment*. Fatalement, il ne peut pas y avoir de drapeau *Don't Fragment*. Les champs réservés doivent rester à 0.

Du reste, c'est comme IPv4 pour le réassemblage. 🍌

#### IV.8.4. ICMP, l'ange gardien

Nous l'avons évoqué à plusieurs reprises, il est temps de se pencher dessus ! Ce fameux ICMP, pour *Internet Control Message Protocol*, est un protocole de niveau 3, c'est-à-dire de la couche réseau. Il fonctionne directement sur IP, sans protocole de transport. Son rôle est de transmettre des messages d'information et d'erreur liés directement au réseau.

Un des cas que peut prendre en charge ICMP et que nous avons vu juste avant, c'est le problème de fragmentation. Parmi les autres cas courants pris en charge, il y a la congestion, un destinataire inaccessible, un problème de routage, ou encore un TTL expiré (tombé à 0).

Pour ce faire, ICMP émet un message très simple, composé de deux numéros. Le premier est le **type** d'erreur ou de demande : destinataire inaccessible (3), TTL expiré (11)... Le deuxième est le **code**, il vient apporter une précision selon le type. Avec un message de type 3, c'est-à-dire de destination injoignable, le code 0 précise que le réseau est inaccessible, le code 4 indique que la fragmentation est nécessaire mais impossible à cause du drapeau DF, etc. La liste est longue et pas forcément utile à connaître par cœur : certains cas sont rarissimes, d'autres sont obsolètes. Vous pouvez consulter tous les types et codes [pour IPv4](#) et [pour IPv6](#) sur le site de l'[IANA](#).

i

Pour indiquer aux hôtes qu'il s'agit d'un message ICMP, l'en-tête IP prend dans son champ « protocole » la valeur 1.

Si on veut être exhaustif, un message ICMP comporte aussi une somme de contrôle ainsi que, de manière facultative, des données additionnelles.

0 0 0 0 0 0 0 0 set tet								1								2								3															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31								
Type								Code								Somme de contrôle																							
Don- nées																																							
(taille																																							

TABLE IV.8.9. – Structure d'un paquet ICMP

Les paquets ICMP pour IPv6 (on abrège en ICMPv6) sont structurés de manière identique à ceux pour IPv4.

Une utilisation très pratique et visuelle d'ICMP est l'écho. Cela consiste à envoyer un petit paquet ICMP avec quelques données à un destinataire, qui est censé nous répondre la même chose. Ça sert à savoir, par exemple, si un hôte est présent sur le réseau, ou bien à mesurer le temps de transit aller-retour entre deux hôtes. Cela vous dit quelque chose ? Eh bien, l'écho ICMP est ce qu'on appelle vulgairement un ping. 🍊

En réalité, **ping** est le nom donné à l'outil permettant d'exploiter ces messages. Ce petit programme est normalement présent sur tous les systèmes. Ouvrez votre console et tapez la commande suivante : **ping zestedesavoir.com**. Si des lignes continuent à défiler après plusieurs secondes, arrêtez le programme avec la commande clavier **CTRL** + **C** (ou **⌘** + **C** sous Mac).

Vous aurez un résultat semblable au suivant (si Zeste de Savoir n'est pas en rade à ce moment précis 🍊 ) :

```
1 Envoi d'une requête 'ping' sur zestedesavoir.com
  [2001:4b98:dc0:41:216:3eff:febc:7e10] avec 32 octets de
  données :
2 Réponse de 2001:4b98:dc0:41:216:3eff:febc:7e10 : temps=11 ms
3 Réponse de 2001:4b98:dc0:41:216:3eff:febc:7e10 : temps=11 ms
4 Réponse de 2001:4b98:dc0:41:216:3eff:febc:7e10 : temps=10 ms
5 Réponse de 2001:4b98:dc0:41:216:3eff:febc:7e10 : temps=10 ms
6
7 Statistiques Ping pour 2001:4b98:dc0:41:216:3eff:febc:7e10:
8   Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
9   Durée approximative des boucles en millisecondes :
10    Minimum = 10ms, Maximum = 11ms, Moyenne = 10ms
```

On peut voir que 4 paquets ont été envoyés. Il s'agit uniquement de requêtes écho ICMP (*echo request*). Le serveur a répondu avec des réponses écho (*echo reply*). Avec des paquets aussi simples, on peut savoir que le serveur est en ligne et que le temps de transit aller-retour moyen est de 10 millisecondes.



Le temps est calculé par le programme ping, le protocole ICMP ne joue pas de rôle dans ces calculs. De même, la récupération de l'adresse IP n'a rien à voir avec ICMP.



Certains réseaux, notamment en entreprise, bloquent les requêtes ICMP pour des raisons de sécurité. Si vous êtes au boulot et que le ping n'a pas marché, vous pouvez réessayer depuis chez vous.

Avant de terminer ce chapitre, vous devez savoir quelque chose... Nous vous avons déjà fait utiliser ICMP dans ce cours. Eh oui ! Dans [Le routage par l'exemple](#), nous vous avons fait faire un **traceroute**. Figurez-vous que ce programme ne fait qu'émettre des requêtes écho !



Mais comment cela peut permettre de déterminer les routeurs intermédiaires ? 🍊

ICMP gère le cas où le TTL est dépassé. Pour rappel, cette valeur fait partie de l'en-tête IP et indique le nombre maximum de sauts autorisés. Quand cette valeur atteint 0, le paquet ne peut plus être routé et il est jeté par le routeur. Mais avant cela, ce dernier envoie un message ICMP à l'expéditeur, histoire qu'il soit quand même au courant.

Vous ne voyez pas où on veut en venir ? Ok, supposons qu'on fasse un **ping zestedesavoir.com** mais en forçant le champ TTL à 1 niveau IP. Que va-t-il se passer ? Le paquet va arriver au premier routeur. Il retire 1 au TTL, qui tombe à 0. Ne pouvant plus router ce paquet, il envoie un message d'erreur à l'expéditeur. Niveau IP, ce message a pour adresse source ce premier routeur, et pour destination, l'émetteur du ping.

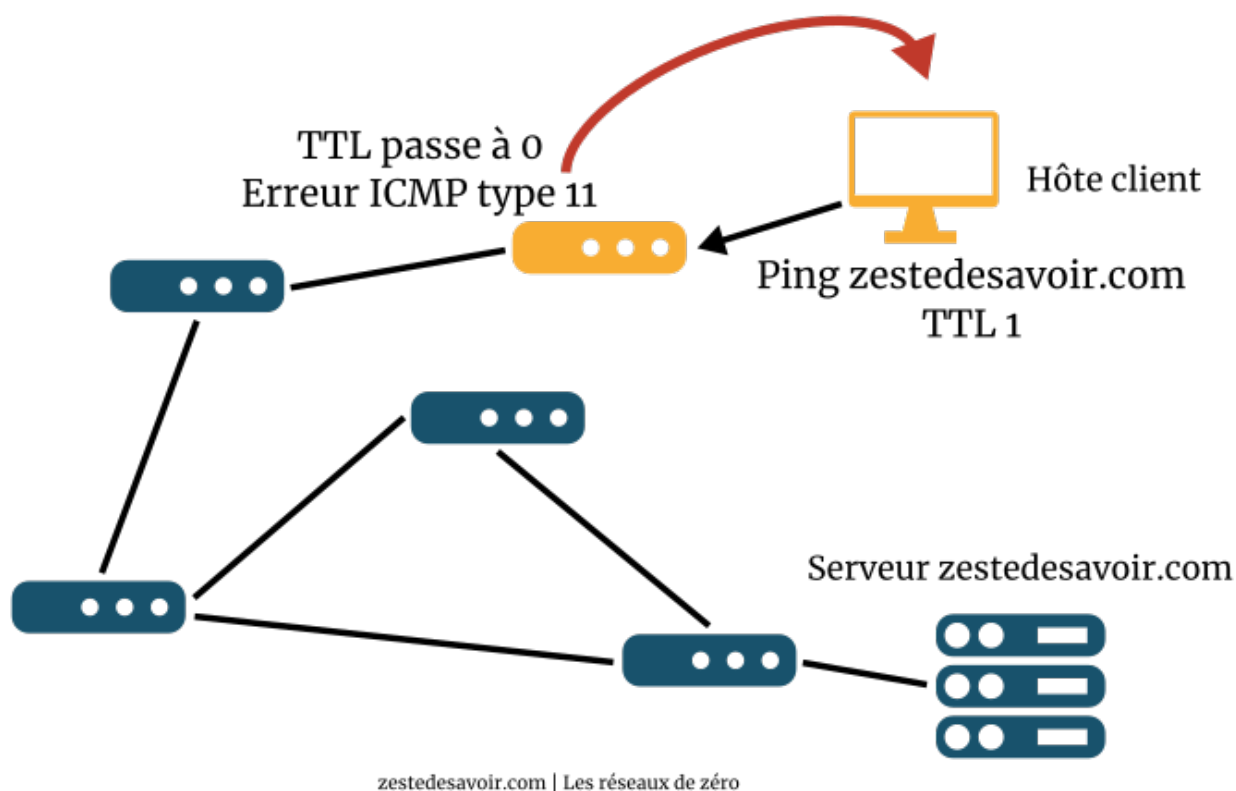


FIGURE IV.8.4. – Envoi d'un ping avec un TTL de 1 (CC BY)

Faisons la même chose en forçant le TTL à 2. On aura un message d'erreur du deuxième routeur. Adresse source : le deuxième routeur, adresse de destination : l'émetteur.

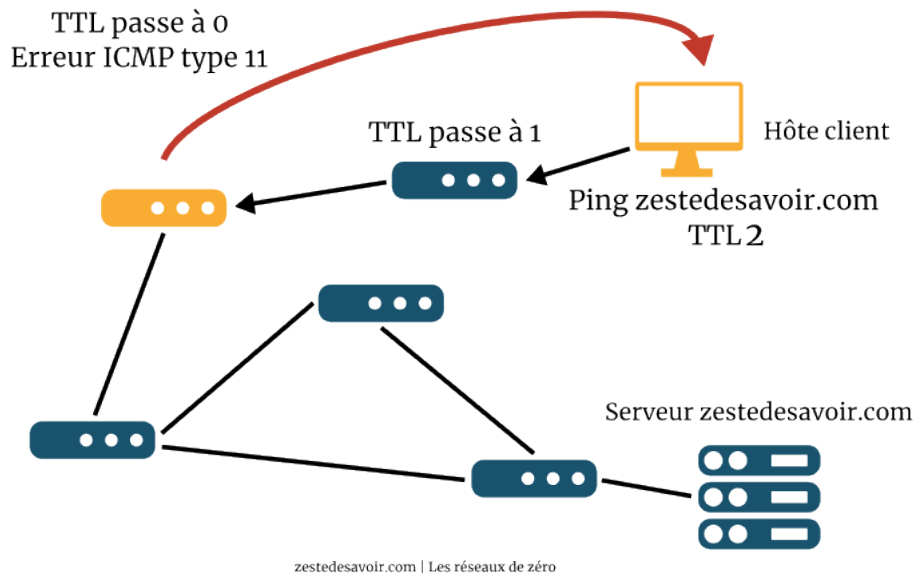


FIGURE IV.8.5. – Envoi d'un ping avec un TTL de 2 (CC BY)

Si on continue ainsi en augmentant à chaque fois le TTL, l'émetteur du ping va recevoir un message d'erreur de chaque routeur, jusqu'à ce qu'il n'y ait plus d'erreur. Il peut donc lister tous les routeurs intermédiaires. C'est exactement ce que fait **tracert** ! Ce programme fait exprès de provoquer des erreurs pour générer des messages ICMP et ainsi déterminer par où passent les paquets.



Les tables de routage peuvent varier subitement sans prévenir, surtout sur Internet. Il est possible que, pour un même **tracert**, les paquets empruntent des chemins différents. Vous n'obtiendrez peut-être pas le même résultat pour un même test selon les jours voire selon les heures. De plus, certains routeurs peuvent ne pas émettre de message d'erreur ICMP : on verra alors des étoiles dans le retour du **tracert**.

Nous nous arrêtons là pour ce protocole qui, avec seulement deux valeurs, rend bien des services !



## Conclusion

IP est incontournable dans le domaine des réseaux depuis des décennies. Tout l'écosystème d'Internet tourne autour de lui. En revanche, pour assurer la communication d'un point de vue physique, on a le choix ! Ce sera le sujet de la prochaine partie.

# Conclusion

Cette partie très chargée vous a permis de comprendre précisément les mécanismes de la communication de bout-en-bout entre hôtes. Maintenant, nous allons passer à la communication entre équipements physiquement reliés. 🍊

# Cinquième partie


## On touche le fond !

# Introduction

Nous y voilà, on arrive aux couches les plus basses du modèle OSI. On va parler des communications entre éléments proches sur le plan logique et physique.

# V.1. Les liaisons dangereuses

## Introduction

Après avoir étudié des mécanismes qui permettent la communication sur, potentiellement, de longues distances, il est temps de se demander comment communiquer avec son proche environnement. Le rôle de la couche 2 est de définir la manière de transmettre des informations à ses voisins. Une petite révision sur [le matériel et les médias d'accès](#)  peut être utile avant de poursuivre. 🍊

### V.1.1. Données, données, do-o-nnées...

Au début de ce cours, nous avons vu différents supports de communication, comme l'air ou le câble de cuivre. Si nous envoyions des paquets IP dans l'air, en Wi-Fi par exemple, nous aurions un problème : parmi tous les appareils du coin, aucun ne saurait lesquels il doit traiter ! Certes, il y a les adresses IP qui permettent de savoir à qui ils sont adressés, mais bon nombre de réseaux utilisent les mêmes adresses privées. Concrètement, la plupart des box Wi-Fi autour de nous ont pour adresse 192.168.1.1 ou 192.168.0.254. 🍊 Il faut une couche intermédiaire pour réguler les communications entre voisins : dans le modèle OSI, elle s'appelle **liaison de données**.

*i*

La notion de voisin a déjà été évoquée précédemment. Elle désigne un hôte directement connecté, ou du moins, perçu comme tel.

La couche 2 a pour rôle principal d'acheminer des **trames** à bon port dans un environnement physique défini. Pour cela, elle a recours à des identificateurs pour reconnaître les équipements ou pour savoir vers où orienter les trames. Un de ces identificateurs est l'adresse MAC, dite aussi adresse physique. On la retrouve notamment dans les réseaux Ethernet, Wi-Fi ou encore Bluetooth.

*i*

Cette notion d'adresse MAC a déjà été abordée plus tôt dans ce cours. Pour rappel, elle correspond à un identifiant unique, inscrit en dur dans une carte réseau.

On peut distinguer deux cas de figure avec la couche 2. Soit on a besoin de différencier les différents hôtes pour savoir qui s'adresse à qui, c'est le cas entre autres du Wi-Fi. Soit il n'y a pas de confusion possible, car le support ne permet que la liaison directe entre deux équipements, c'est le cas pour l'ADSL. Par la suite, nous étudierons en détail au moins un protocole de chaque catégorie.

## V.1.2. Les interfaces

Il est courant qu'une machine dispose de plusieurs cartes réseaux, qui sont autant d'**interfaces physiques**. Une interface, globalement, est un point d'entrée ou de sortie d'une communication. Par exemple, sur une carte réseau Ethernet, le trou dans lequel on branche un câble, c'est une interface.

Votre ordinateur, ou plus exactement les programmes que vous utilisez, n'ont pas conscience de l'aspect matériel du réseau. Le système d'exploitation fournit des **interfaces logiques** qui permettent d'émettre des trames. Après, que ce soit émis sur des fils de cuivre, une fibre optique ou autre, peu importe ! 🍊

On peut visualiser les interfaces logiques depuis la console avec la commande `ipconfig /all` sous Windows ou `ifconfig -a` sous Linux<sup>1</sup>. Vous obtiendrez un résultat similaire au suivant :

```
1 Configuration IP de Windows
2
3 Nom de l'hôte . . . . . : XXX
4 Suffixe DNS principal . . . . . :
5 Type de noeud. . . . . : Hybride
6 Routage IP activé . . . . . : Non
7 Proxy WINS activé . . . . . : Non
8 Liste de recherche du suffixe DNS.: home
9
10 Carte Ethernet :
11
12 Statut du média. . . . . : Média déconnecté
13 Suffixe DNS propre à la connexion. . . :
14 Description. . . . . : Realtek PCIe GBE
    Family Controller
15 Adresse physique . . . . . : 01-23-45-56-78-9A
16 DHCP activé. . . . . : Oui
17 Configuration automatique activée. . . : Oui
18
19 Carte Ethernet Npcap Loopback Adapter :
20
21 Suffixe DNS propre à la connexion. . . :
22 Description. . . . . : Npcap Loopback Adapter
23 Adresse physique . . . . . : 02-03-04-05-06-07
24 DHCP activé. . . . . : Oui
25 Configuration automatique activée. . . : Oui
26 Adresse IPv6 de liaison locale. . . . . :
    fe80::1234:5678:90ab:cdef%19(préfér  )
27 Adresse d'autoconfiguration IPv4 . . . : 169.254.42.42(pr  f  r  )
28 Masque de sous-r  seau. . . . . : 255.255.0.0
29 Passerelle par d  faut. . . . . :
30 IAID DHCPv6 . . . . . : 218234272
31 DUID de client DHCPv6. . . . . :
    00-01-00-01-12-34-56-78-90-AB-CD-EF-00-00
32 Serveurs DNS. . . . . : fec0:0:0:ffff::1%1
```

## V. On touche le fond !

```

33                                     fec0:0:0:ffff::2%1
34                                     fec0:0:0:ffff::3%1
35     NetBIOS sur Tcpip. . . . . : Activé
36
37 Carte réseau sans fil Connexion au réseau local* 1 :
38
39     Statut du média. . . . . : Média déconnecté
40     Suffixe DNS propre à la connexion. . . :
41     Description. . . . . : Microsoft Wi-Fi Direct
42     Virtual Adapter
43     Adresse physique . . . . . : 32-10-98-76-54-32
44     DHCP activé. . . . . : Oui
45     Configuration automatique activée. . . : Oui
46
47 Carte réseau sans fil Connexion au réseau local* 2 :
48
49     Statut du média. . . . . : Média déconnecté
50     Suffixe DNS propre à la connexion. . . :
51     Description. . . . . : Microsoft Wi-Fi Direct
52     Virtual Adapter #2
53     Adresse physique . . . . . : 32-10-98-76-54-31
54     DHCP activé. . . . . : Oui
55     Configuration automatique activée. . . : Oui
56
57 Carte réseau sans fil Wi-Fi :
58
59     Suffixe DNS propre à la connexion. . . : blah
60     Description. . . . . : Intel(R) Wireless-AC
61     XXXX
62     Adresse physique . . . . . : 76-54-32-32-10-98
63     DHCP activé. . . . . : Oui
64     Configuration automatique activée. . . : Oui
65     Adresse IPv6. . . . . :
66     2a01:0123:456:ff00:1234:5678:90ab:cdef(préfééré)
67     Adresse IPv6 temporaire . . . . . :
68     2a01:0123:456:ff00:90ab:cdef:1234:5678(préfééré)
69     Adresse IPv6 de liaison locale. . . . :
70     fe80::0123:456:0123:456%15(préfééré)
71     Adresse IPv4. . . . . : 192.168.1.10(préfééré)
72     Masque de sous-réseau. . . . . : 255.255.255.0
73     Bail obtenu. . . . . : jeudi 7 février 2019
74     12:00:00
75     Bail expirant. . . . . : mercredi 13 février
76     2019 18:00:00
77     Passerelle par défaut. . . . . :
78     fe80::456:0123:456:0123%15
79
80     192.168.1.1
81     Serveur DHCP . . . . . : 192.168.1.1
82     IAID DHCPv6 . . . . . : 112233777

```

## V. On touche le fond !

```
73 DUID de client DHCPv6. . . . . :  
    00-01-00-01-23-45-67-89-0A-BC-DE-F0-00-00  
74 Serveurs DNS. . . . . :  
    fe80::b2b2:8fff:fe73:c05a%15  
75                                     192.168.1.1  
76                                     fe80::456:0123:456:0123%15  
77 NetBIOS sur Tcpip. . . . . : Activé  
78 Liste de recherche de suffixes DNS propres à la connexion :  
79                                     home  
80                                     home  
81  
82 Carte Ethernet Connexion réseau Bluetooth :  
83  
84 Statut du média. . . . . : Média déconnecté  
85 Suffixe DNS propre à la connexion. . . :  
86 Description. . . . . : Bluetooth Device  
    (Personal Area Network)  
87 Adresse physique . . . . . : 88-88-88-88-88-88  
88 DHCP activé. . . . . : Oui  
89 Configuration automatique activée. . . : Oui
```

Ne vous formalisez pas trop sur les noms des interfaces, c'est les bizarreries de Windows... 🍊  
Sous Linux, c'est plus sobre :

```
1 eth0      Link encap:Ethernet  HWaddr 09:00:12:90:e3:e5  
2           inet addr:192.168.1.29 Bcast:192.168.1.255  
           Mask:255.255.255.0  
3           inet6 addr: fe80::a00:27ff:fe70:e3f5/64 Scope:Link  
4           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
5           RX packets:54071 errors:1 dropped:0 overruns:0 frame:0  
6           TX packets:48515 errors:0 dropped:0 overruns:0 carrier:0  
7           collisions:0 txqueuelen:1000  
8           RX bytes:22009423 (20.9 MiB)  TX bytes:25690847 (24.5  
           MiB)  
9           Interrupt:10 Base address:0xd020  
10  
11 lo       Link encap:Local Loopback  
12           inet addr:127.0.0.1  Mask:255.0.0.0  
13           inet6 addr: ::1/128 Scope:Host  
14           UP LOOPBACK RUNNING  MTU:16436  Metric:1  
15           RX packets:83 errors:0 dropped:0 overruns:0 frame:0  
16           TX packets:83 errors:0 dropped:0 overruns:0 carrier:0  
17           collisions:0 txqueuelen:0  
18           RX bytes:7766 (7.5 KiB)  TX bytes:7766 (7.5 KiB)  
19  
20 wlan0     Link encap:Ethernet  HWaddr 58:a2:c2:93:27:36  
21           inet addr:192.168.1.64  Bcast:192.168.2.255  
           Mask:255.255.255.0
```

## V. On touche le fond !

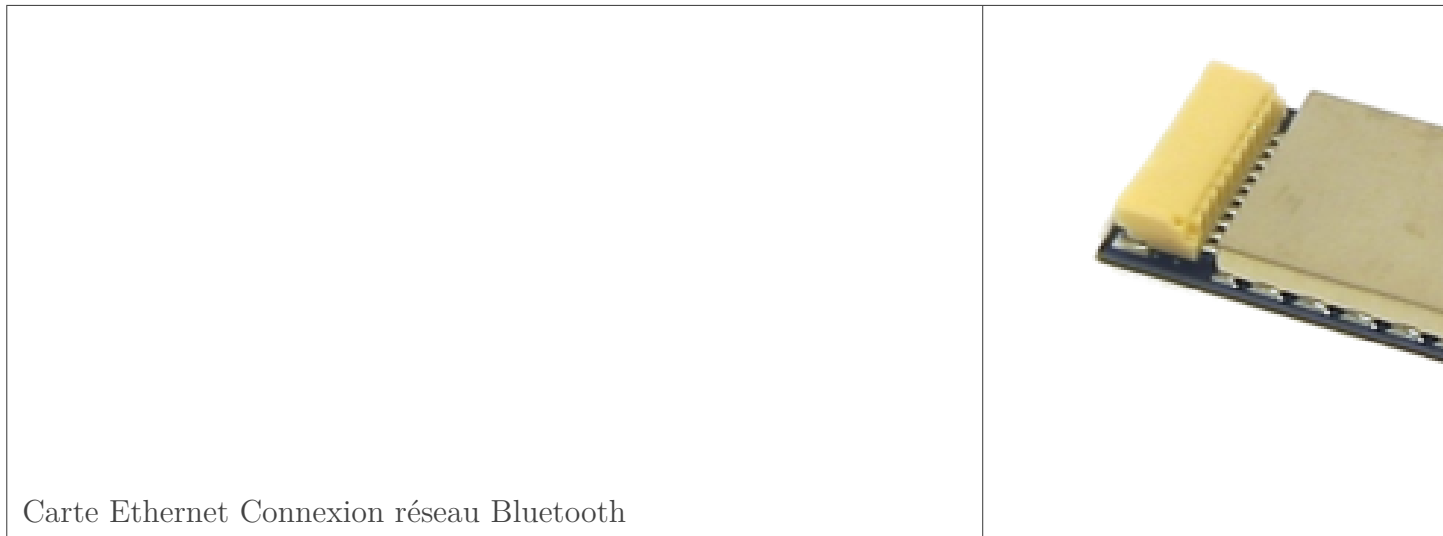
22	inet6 addr: fe80::6aa3:c4ff:fe93:4746/64 Scope:Link
23	UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
24	RX packets:436968 errors:0 dropped:0 overruns:0 frame:0
25	TX packets:364103 errors:0 dropped:0 overruns:0 carrier:0
26	collisions:0 txqueuelen:1000
27	RX bytes:115886055 (110.5 MiB) TX bytes:83286188 (79.4 MiB)

Ce dernier extrait est issu de [cette page de Computer Hope](#) .

Certains termes doivent vous être familiers ! 🍏 On peut faire le lien suivant entre interfaces logiques et physiques :

Interface logique	Interface physique
Carte réseau sans fil Wi-Fi / wlan0	
Carte Ethernet / eth0	

## V. On touche le fond !



Crédits images :

- carte Wi-Fi dérivée de [ce travail d'AndreR](#) sous licence GNU GPL 2, image redistribuée sous licence GNU GPL 2 ;
- carte Ethernet dérivée de [cette photo par Sub](#) dans le domaine public, redistribuée sous licence CC BY-NC-ND 2.0 ;
- carte Bluetooth dérivée de [cette photo par Omegatron](#) sous licence CC BY SA, redistribuée sous licence CC BY SA 3.0.

Mais ce n'est pas tout ! Plusieurs interfaces logiques peuvent en réalité correspondre à une seule interface physique. Mieux encore : une interface logique peut ne correspondre à aucune réalité physique.

?

Quoi ? Mais pourquoi on aurait une interface qui ne sert pas à communiquer ? 🍊

Vous vous souvenez des adresses IP comme 127.0.0.1 ou ::1 ? Eh bien, elles sont attribuées à une interface nommée **boucle locale** ou *loopback*. C'est l'interface **lo** dans l'extrait de configuration Linux précédent. Elle sert à s'envoyer des paquets à soi-même, donc pas besoin d'interface physique, puisque ça reste au sein du même système. 🍊

## Conclusion

Voilà pour une présentation assez générale et succincte de la couche 2. Elle permet la communication entre voisins au travers d'interfaces. Les possibilités sont assez vastes et souvent liées à un protocole donné. Aussi, nous étudierons des services de la couche liaison de données avec un protocole associé.

---

1. Cette commande n'est plus présente par défaut sous certaines distributions Linux, qui lui préfèrent la commande **ip addr**. Toutefois, nous trouvons que **ifconfig** affiche des informations plus intéressantes et mieux présentées. Vous pouvez l'installer avec le paquet *net-tools*.

## V.2. Qu'est-ce qui se trame chez Ethernet ?

### Introduction

Ethernet, voilà un terme qu'on entend très souvent en réseau. Il désigne plusieurs choses à la fois. Nous allons d'abord voir comment se présente un réseau Ethernet physiquement, puis nous nous focaliserons sur le protocole de niveau 2 associé.

#### V.2.1. Un réseau Ethernet, kézako ?

Un réseau Ethernet, ça peut se résumer ainsi : un ensemble de machines reliées sur un switch par des câbles Ethernet. Le switch, ou commutateur, est la pièce maîtresse ici. Tous les hôtes sont branchés dessus selon une topologie en étoile.

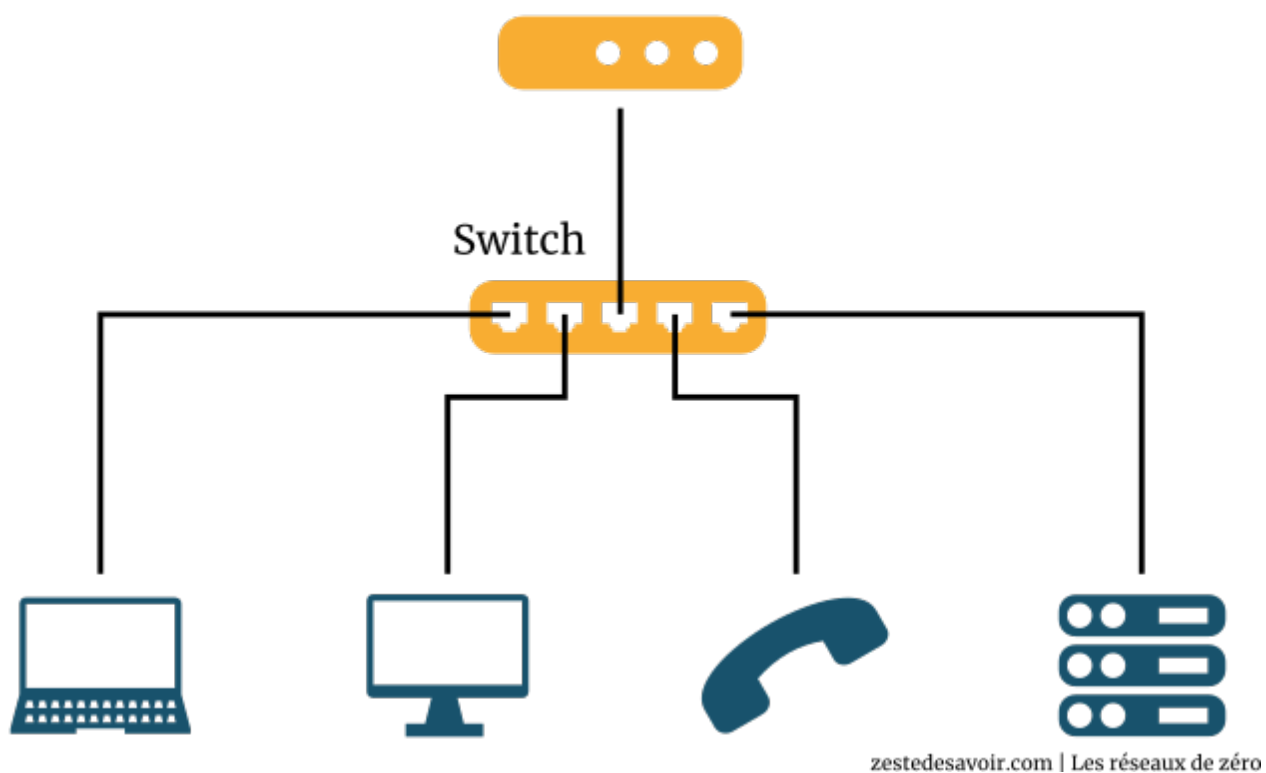


FIGURE V.2.1. – Réseau Ethernet basique (CC BY)

i

Il s'agit là du cas le plus simple. Un réseau Ethernet peut aussi être beaucoup plus complexe.

## V. On touche le fond !

Lorsqu'un hôte veut émettre des données, il les envoie sur le câble et elles sont reçues par le switch. Ce dernier analyse la trame et procède éventuellement à des vérifications : est-ce que cet hôte est bien autorisé à communiquer sur cette interface ? La destination est-elle un broadcast ? Un unicast ? Est-ce que je la connais, au moins ? Puis, si tout va bien, le switch renvoie la trame au bon destinataire. En résumé, le commutateur vérifie (ou non) la légitimité de l'émetteur et transmet les données au destinataire.

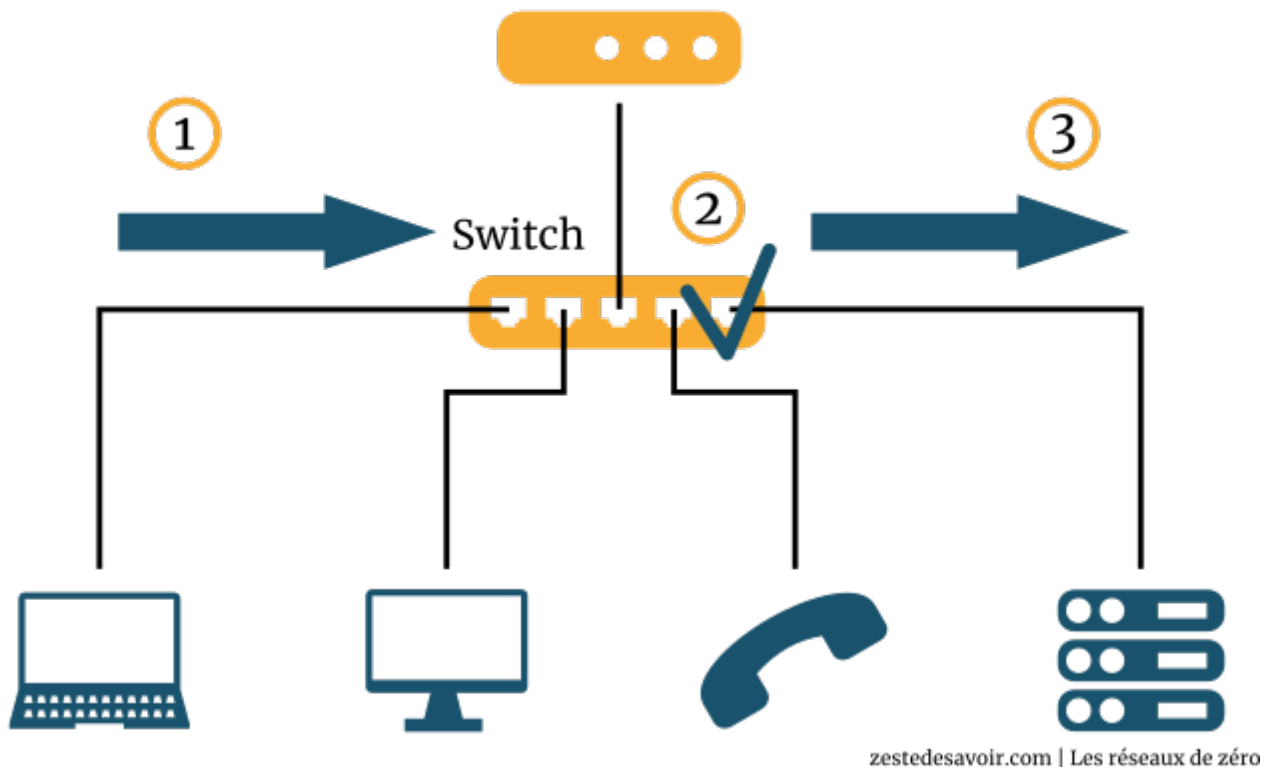


FIGURE V.2.2. – Étapes de commutation (CC BY)

Évidemment, il y a quelques petites subtilités qui rendent la chose un poil plus complexe. 🍊 Étudions d'abord le protocole Ethernet qui est ultra simple dans le principe... et peut générer des problèmes monstrueux en pratique. 🤨

## V.2.2. Le protocole

Sur un réseau Ethernet, on utilise bien souvent... le protocole Ethernet. Voyons la structure d'une trame Ethernet.

0000							1							2							3																
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	1200	1201	1210	1211	1220	1221	1230	1231	1240	1241	1250	1251	1260	1261	1270	1271	1280	1281	1290	1291	1300	1301

V. On touche le fond !

0	0	Adresse MAC des- ti- na- tion																												
4	32	Adresse MAC des- ti- na- tion (suite)	Adresse MAC source																											
8	64	Adresse MAC source (suite)																												
12	96	Ether- Type (si > 1500 uni- que- ment)	Don- nées																											
16	128																													
20	160																													
24	192																													
28	224																													
32	256																													
36	288	Don- nées (suite)																												
40	320																													
44	352																													
48	384																													
52	416																													
56	448																													
60	480																													
...	...																													

## V. On touche le fond !

		Somme
...	...	de
		contrôle
		(FCS)

C'est assez simple : adresses MAC source et destination, EtherType, données, somme de contrôle. Une trame, hors somme de contrôle, doit faire au minimum 64 octets. Si ça fait moins, eh bien, on rajoute des zéros après les données jusqu'à ce que ça fasse assez : on appelle ça du **bourrage**. La chose intéressante, c'est l'**EtherType**.

Ce champ, sur deux octets, est un peu particulier. Déjà, sa signification n'est pas la même selon que sa valeur est supérieure ou non à  $1500_{(10)}$  ( $05DC_{16}$ ). Pour des questions de compatibilité, une valeur de 1500 ou moins indique la longueur des données à suivre. Dans le cas contraire, qui nous intéresse, il définit quel protocole de niveau supérieur va suivre, pour que le récepteur sache à quoi s'attendre. C'est un peu comme le champ "protocole" avec IPv4. Par exemple, la valeur  $0800_{16}$  correspond à IPv4,  $86DD_{16}$  à IPv6. Mais il y a une valeur particulière :  $8100_{16}$ . Celle-ci précise l'appartenance à un **VLAN**.

### V.2.3. Et VLAN !

Un VLAN, pour *Virtual LAN*, ou réseau local virtuel, c'est un moyen de segmenter de manière logique un réseau Ethernet. Vous vous souvenez que les communications passent par un switch ? Ce dernier peut créer virtuellement des sous-réseaux étanches, qui ne peuvent pas communiquer entre eux. Pour cela, il colle une étiquette (ou un *tag*) avec un numéro sur certains ports. Ainsi, si les interfaces 1 et 2 sont étiquetées "VLAN 21", et les interfaces 3 et 4 "VLAN 42", pas moyen de communiquer entre l'hôte du port 1 et celui du port 3, pas plus qu'entre celui du port 2 et celui du port 4 ! Cette technique permet d'empêcher que dans un LAN, des machines ne communiquent avec d'autres alors qu'elles ne devraient rien avoir à se dire. Un téléphone fixe qui voudrait communiquer avec un PC de bureau, ce serait quand même louche, avouez ! 🍊

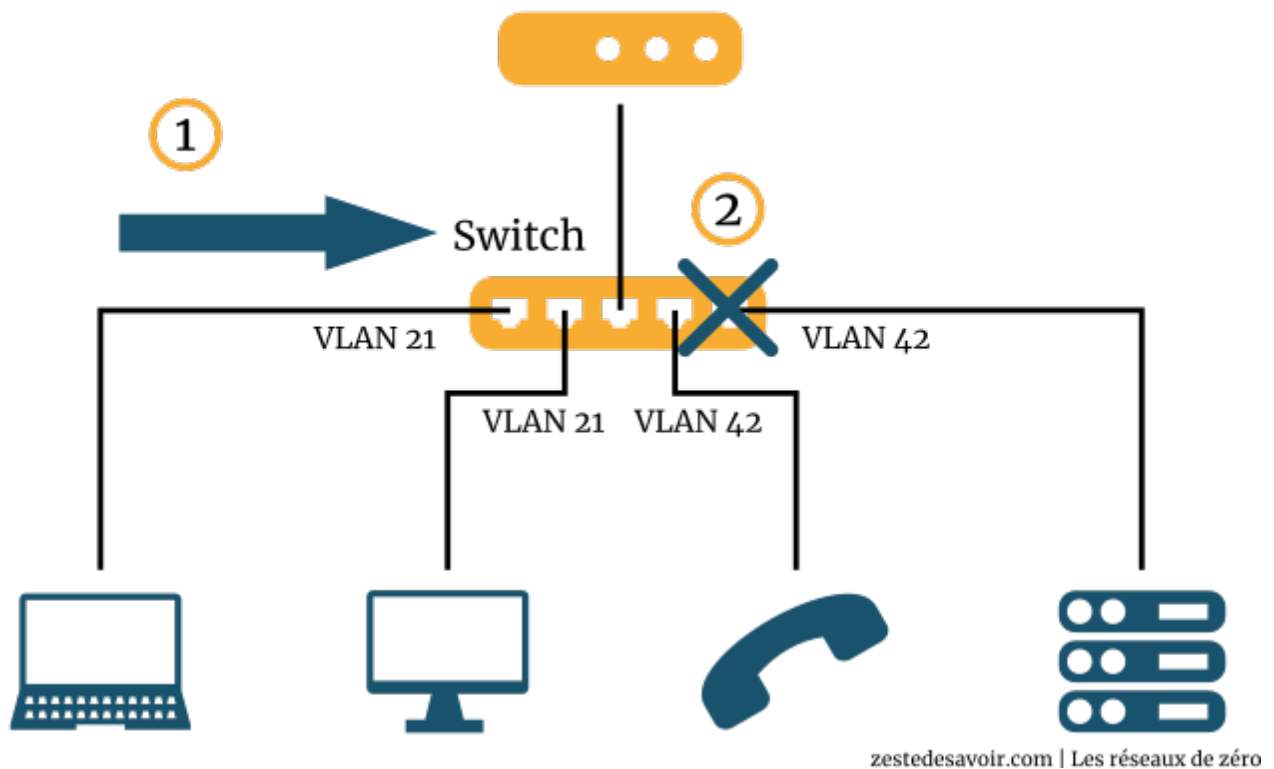


FIGURE V.2.3. – Commutation impossible entre 2 VLAN différents (CC BY)

i

Un VLAN peut prendre un numéro entre 2 et 4094, à l'exception des valeurs entre 1002 et 1005, qui sont réservées. Par convention, le VLAN 1 correspond à "pas de VLAN" : tous les ports d'un switch non tagués (étiquetés) font partie du VLAN 1.

Cela est transparent pour tout le monde, seul le switch est au courant de tout cela. Là où ça se complexifie, c'est quand il y a plusieurs switches reliés entre eux dans le LAN. Les liens qui les relient peuvent transporter des trames en provenance de plusieurs VLAN. Un tel lien est appelé **trunk**. On peut le configurer pour préciser quels VLAN ont le droit de transiter par là. Il faut s'assurer que la configuration est la même de chaque côté du trunk ! Ce genre d'erreur est la source de nombreuses prises de tête, d'heures perdues, de crises, d'apocalypses nucléaires... Encore que s'il y a vraiment eu une apocalypse nucléaire, il y a peu de chances que vous soyez en train de lire un cours de réseaux. 🍊

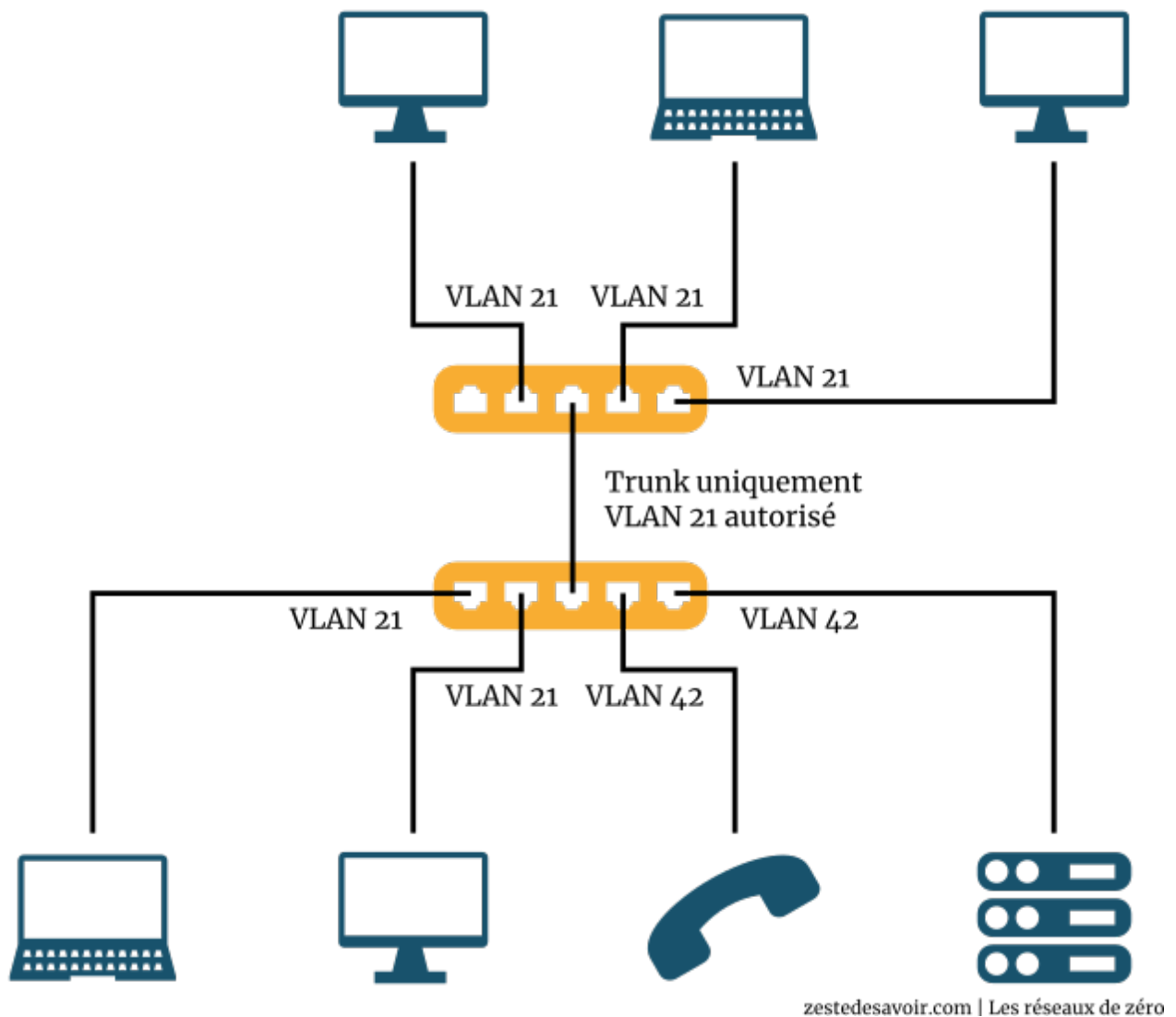


FIGURE V.2.4. – Un lien trunk permet de transporter un ou plusieurs VLAN (CC BY)

Sur un trunk, pour préciser quelle trame va avec quel VLAN, on modifie un peu la trame Ethernet. Le standard IEEE 802.1Q nous indique qu'il faut faire comme suit.

Offset								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Adresse MAC								des- ti- na- tion																							

## V. On touche le fond !

4	32	Adresse MAC des- ti- na- tion (suite)										Adresse MAC source											
8	64	Adresse MAC source (suite)																					
12	96	Ether- Type = 8100 <sub>16</sub>										Prio- rité	CF	VLAN ID									
16	128	Ether- Type réel										Don- nées											
20	160																						
24	192																						
28	224																						
32	256																						
36	288																						
40	320																						
44	352																						
48	384																						
52	416																						
56	448																						
60	480																						
...	...																						
...	...	Somme de contrôle (FCS)																					

3 nouveaux champs font leur apparition avant que l'EtherType réel ne soit précisé. La priorité, sur 3 bits, sert si on veut définir un niveau de priorité entre les VLAN. On peut faire un parallèle avec le DSCP du protocole IP. On passe le bit suivant, il est là pour des raisons de compatibilité avec Token Ring. Les 12 bits qui viennent ensuite correspondent au numéro de VLAN.



Un point prononciation : en français, les administrateurs réseau prononcent VLAN "vé-lane". Pour parler du standard 802.1Q, vous entendrez parfois "dot 1 Q" : *dot* pour point, le chiffre 1 et la lettre Q prononcés à la française. Dans les configurations de switches, on lit parfois le mot "dot1q" et c'est beaucoup plus naturel de le prononcer comme ça. 🍊

Pour conclure cette section, sachez qu'on ne peut pas assigner d'adresse IP à un port (ça semble logique), mais on peut en attribuer aux VLAN. Cela sert notamment pour contrôler le switch à distance. 🍊

## V.2.4. 1, 2, 3, on switche !

Si on ne devait retenir que 3 choses du protocole Ethernet, ce serait :

- Adresse MAC source
- Adresse MAC destination
- VLAN

Ces trois seules notions peuvent donner du fil à retordre au switch (et surtout aux administrateurs réseau). Revenons sur les étapes de commutation évoquées au début de ce chapitre.

### V.2.4.1. Réception

En premier lieu, le switch reçoit une trame sur une interface et regarde l'adresse MAC source. D'abord, il s'assure qu'il n'y a pas de restriction en terme d'émetteur sur ce port. Il peut y avoir une liste d'adresses interdites (les autres étant autorisées), ou une liste d'adresses autorisées (les autres étant interdites). Un autre cas de restriction est le nombre de sources permises en simultané. Si, juste avant, une autre adresse MAC source vient de passer sur ce même port, il y a peut-être anguille sous roche. 🐛 Dans ce cas, on peut bloquer le port, enregistrer l'incident et avertir l'administrateur réseau (#balancetonport) !



Ce cas peut aussi être tout à fait légitime. Si un autre switch est branché sur ce port, il est normal de voir plusieurs adresses sources transiter.

Enfin, dans cette première étape, on regarde si la trame est taguée avec un VLAN. Si c'est le cas, le switch vérifie que ce VLAN est bien autorisé sur ce port. Sinon, il s'assure que les trames non taguées sont admises. Pareillement, en cas d'interdiction rencontrée, les trames sont jetées et l'incident peut être enregistré.

### V.2.4.2. Commutation

Une fois que le switch a déterminé que la trame reçue était légitime, il doit la **commuter**. Cela consiste à la réémettre sur une autre interface.



C'est comme le routage, alors ?

## V. On touche le fond !

Ça y ressemble, mais c'est beaucoup plus simple. Dans le cadre d'un réseau Ethernet, chaque switch dispose d'une table de correspondance interfaces / adresses MAC. Il apprend les associations en regardant les adresses MAC sources des trames légitimes qu'il reçoit. La commutation consiste à transmettre chaque trame au bon destinataire.

?

Si la destination n'est pas connue, il se passe quoi ? 🍊

Ça peut arriver, surtout juste après le démarrage. Dans ce cas... la trame est tout bonnement *broadcastée* sur tous les ports associés au même VLAN que la source. Bonjour la confidentialité !



i

Il y a un autre cas où cela peut arriver : lors de multicast. Une adresse MAC multicast commence par 01:00:5E (avec IPv4) ou 33:33 (avec IPv6). Certains switchs savent les gérer et retransmettre sur les bonnes interfaces. Mais les moins avancés se contentent de broadcaster !

### V.2.4.3. Transmission

Une fois la décision prise pour la commutation, la trame est transmise sur le ou les ports concernés, à condition que la source et la destination soient dans le même VLAN. Selon la configuration, un tag de VLAN peut être ajouté (transmission de la trame à un autre switch ou routeur, trunk) ou retiré (transmission à un hôte qui n'a pas conscience de son VLAN). Éventuellement, il peut y avoir des logs ou des enregistrements réalisés à des fins d'analyse.

Il existe quelques subtilités pour cette dernière étape. Par exemple, certains switchs permettent de retransmettre sur un port défini tout ce qui arrive sur une autre interface, indépendamment de la source ou de la destination. Cette fonctionnalité est appelée *port mirroring*. Nous n'allons pas nous étendre dessus, sachez juste que ça existe. 🍊

## Conclusion

Nous avons fait un petit tour des réseaux Ethernet en présentant leur architecture, le protocole de niveau 2 associé et le commutateur. Par la suite, nous continuerons à explorer la couche liaison de données en étudiant d'autres genres de protocoles. 🍊

Si vous souhaitez étudier le Wi-Fi au niveau 2, nous vous recommandons [cet article \(archivé\) de CommentÇaMarche](#) 🍊 . Vous y trouverez même une pointe de niveau 1. 🍊

Enfin, vous repenserez à ce chapitre si, en vous baladant à Toulouse, vous vous retrouvez [rue Vélane](#) 🍊 .

## V.3. Deux points c'est tout

### Introduction

Au début de cette partie sur la couche 2, nous avons évoqué le fait qu'un protocole de liaison de données peut servir même lorsqu'il n'y a qu'une source et une destination possibles. Nous allons en premier lieu étudier l'un de ces protocoles dans un cas simple, que nous complexifierons de plus en plus. 🍊

#### V.3.1. Tout vient à point...

Vous avez déjà entendu parler de l'ADSL ? C'est une technologie très utilisée dans les années 2000 - 2010 notamment pour l'accès à Internet chez les particuliers. Cela tend à disparaître depuis les années 2020 et la généralisation de la fibre optique, mais c'est le système parfait pour illustrer le propos de ce chapitre, alors on se calme ! 🍊 Physiquement, en ADSL, une paire de fils de cuivre va de chez l'abonné aux équipements de l'opérateur. Pas de switch, pas de routeur entre l'abonné et l'opérateur : c'est une liaison directe.



On peut envoyer directement des paquets IP, alors ?

Eh bien... non. Un des problèmes qui va se poser, c'est l'authentification de l'utilisateur. L'opérateur pourrait techniquement l'identifier par la ligne, mais comment être sûr que c'est bien lui et non une personne malveillante qui tente d'utiliser cette ligne incognito ?

Pour cela, on peut avoir recours à **PPP** : Point-to-Point Protocol (protocole point-à-point). Ce protocole de liaison de données sert à établir un lien de niveau 2 entre deux entités, et surtout, permet l'authentification. Ainsi, dans le cas de notre ligne ADSL, on va pouvoir s'assurer qu'un méchant pirate n'essaie pas de détourner une ligne de cuivre pour commettre des malversations.



#### Authentification et identification

Ces termes vous semblent compliqués ? [Par ici pour une petite explication ↗](#) !




Théoriquement, PPP peut être utilisé sans authentification. En pratique, ce cas est rare et peu utile.

Deux méthodes d'authentification sont supportées par PPP : elles s'appellent PAP et CHAP. Ce protocole fonctionne selon une architecture client-serveur. C'est toujours le client qui initie la connexion et qui choisit son mode d'authentification. Si c'est PAP, c'est très facile : les identifiants sont directement envoyés en clair au serveur dès le départ ! Ce dernier répond qu'ils

## V. On touche le fond !

sont valides ou non, et la session est établie ou non. Simple et efficace, et absolument pas sécurisé ! 🍊 Une personne qui peut voir ce qui se passe sur la ligne récupérerait sans difficulté les mots de passe. Pour éviter cela, mieux vaut utiliser CHAP.

Dans ce dernier cas, cela fonctionne différemment. Le client fait une demande de connexion. Le serveur lui renvoie un **challenge**. Il contient un numéro d'identification, un nombre aléatoire ainsi que le nom du serveur. Pour relever ce défi, le client doit renvoyer le même numéro d'identification, son nom d'utilisateur ainsi qu'un **hash**<sup>1</sup>. Ce hash, ce n'est pas une substance qui se fume et qui est très mauvaise pour la santé ! C'est une valeur basée sur le numéro d'identification, le nombre aléatoire et le mot de passe de l'utilisateur. Pour plus de détails sur ce procédé, nous vous invitons à consulter [ce document du constructeur Cisco](#) . C'est en anglais et ça parle surtout de configuration de routeurs, mais les schémas sont plutôt clairs.

Tout cela est transmis au serveur. Le hash assure la confidentialité du mot de passe sur le réseau. Le serveur peut faire le même calcul de son côté, puisqu'il connaît d'une manière ou d'une autre le mot de passe associé au nom d'utilisateur. Ainsi, il vérifie si le challenge est réussi : si tel est le cas, bravo ! Le client est récompensé par une connexion PPP établie ! 🍊

PPP est donc bien utile pour fournir de l'authentification sur un lien direct. Mais bien souvent, le support physique ne permet pas de transmettre directement des trames PPP. C'est là que ça se complexifie. 🍊

### V.3.2. ... à qui sait encapsuler

Sur les lignes ADSL, un protocole qu'il est fréquent de rencontrer est **ATM**. Avec lui, chaque envoi de données se fait dans des **cellules**, qui sont des "mini-trames" de taille fixe (53 octets). Chaque cellule comporte un en-tête qui permet de définir des chemins virtuels et des circuits virtuels. Nous ne nous étendrons pas sur ces notions ici, sachez simplement que c'est ce qui permet de différencier ce qui relève du téléphone, de la télévision et d'Internet dans les offres dites "triple play". 🍊

Pour des raisons d'ordre électronique que nous ne développerons pas pour le moment, il n'est pas possible de faire transiter directement du PPP sur de l'ADSL. On peut en revanche utiliser le protocole **ATM**. Comment faire alors pour bénéficier des services de PPP sur une ligne ADSL ? Eh bien... **en encapsulant PPP dans ATM**.

Tout comme TCP peut s'encapsuler dans IP, comme IP peut s'encapsuler dans Ethernet, certains protocoles de niveau 2 peuvent s'encapsuler entre eux. Dans le cas que nous avons évoqué, on parle de PPP over **ATM** (PPPoA). De la même manière, on peut encapsuler PPP dans Ethernet, ce qu'on appelle PPPoE. Ethernet pouvant à son tour être encapsulé dans **ATM**, on peut même faire du PPPoEoA ! 🍊

Ce genre d'encapsulation crée plusieurs interfaces sur votre système. Si vous établissez une liaison PPP entre deux hôtes reliés en Ethernet, vous aurez une interface Ethernet **et** une interface PPP ! Chacune peut être paramétrée indépendamment, notamment en matière d'adressage.

Prenons un exemple : nos 2 hôtes Clémentine et Mandarine sont reliés en Ethernet et ont ouvert une connexion PPP. Elles disposent d'adresses IP configurées ainsi :

-	Clémentine	Mandarine
PPP	10.0.0.1/24	10.0.0.2/24

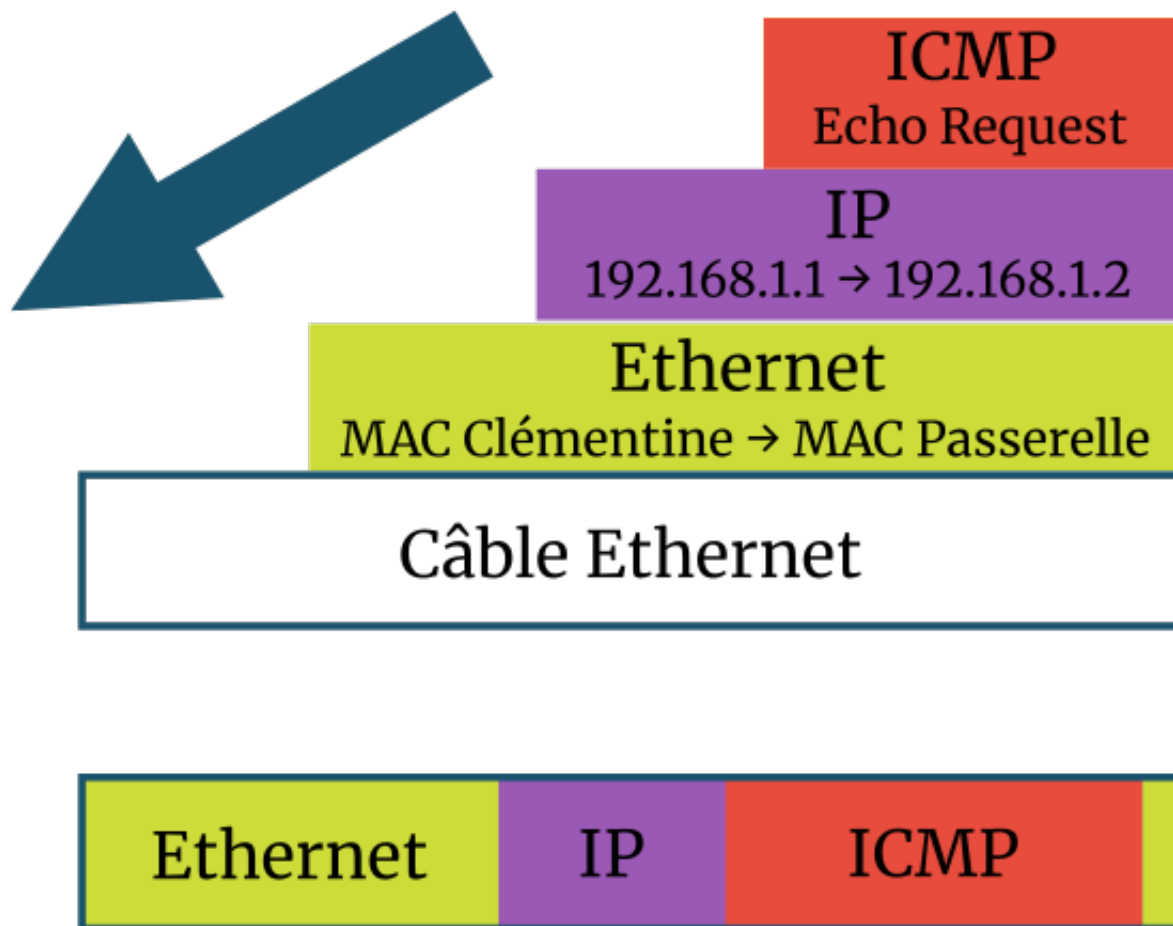
---

1. Le hashage est une sorte de chiffrement à sens unique. On ne peut pas retrouver une valeur d'origine qui a été hashée, mais la même fonction de hashage donne toujours le même résultat pour une même valeur.

## V. On touche le fond !

Ethernet	192.168.1.1/24	192.168.1.2/24
----------	----------------	----------------

Pour communiquer, nos agrumes peuvent passer par 2 réseaux IP. Si Clémentine lance un `ping 192.168.1.2`, que se passe-t-il ? Facile : ce réseau est directement connecté sur l'interface Ethernet. On aura le comportement suivant :



zestedesavoir.com | Les réseaux de zéro

FIGURE V.3.1. – Un paquet simplement encapsulé dans Ethernet (CC BY)

Et si Clémentine fait un `ping 10.0.0.2` ? Ce réseau est directement connecté sur l'interface PPP... Mais celle-ci doit s'appuyer sur Ethernet ! Le programme ping n'en a pas conscience, il ne peut voir que l'interface PPP pour ce réseau. Le système d'exploitation se débrouille pour gérer cela. Sur le câble, voici ce qu'on va voir passer.

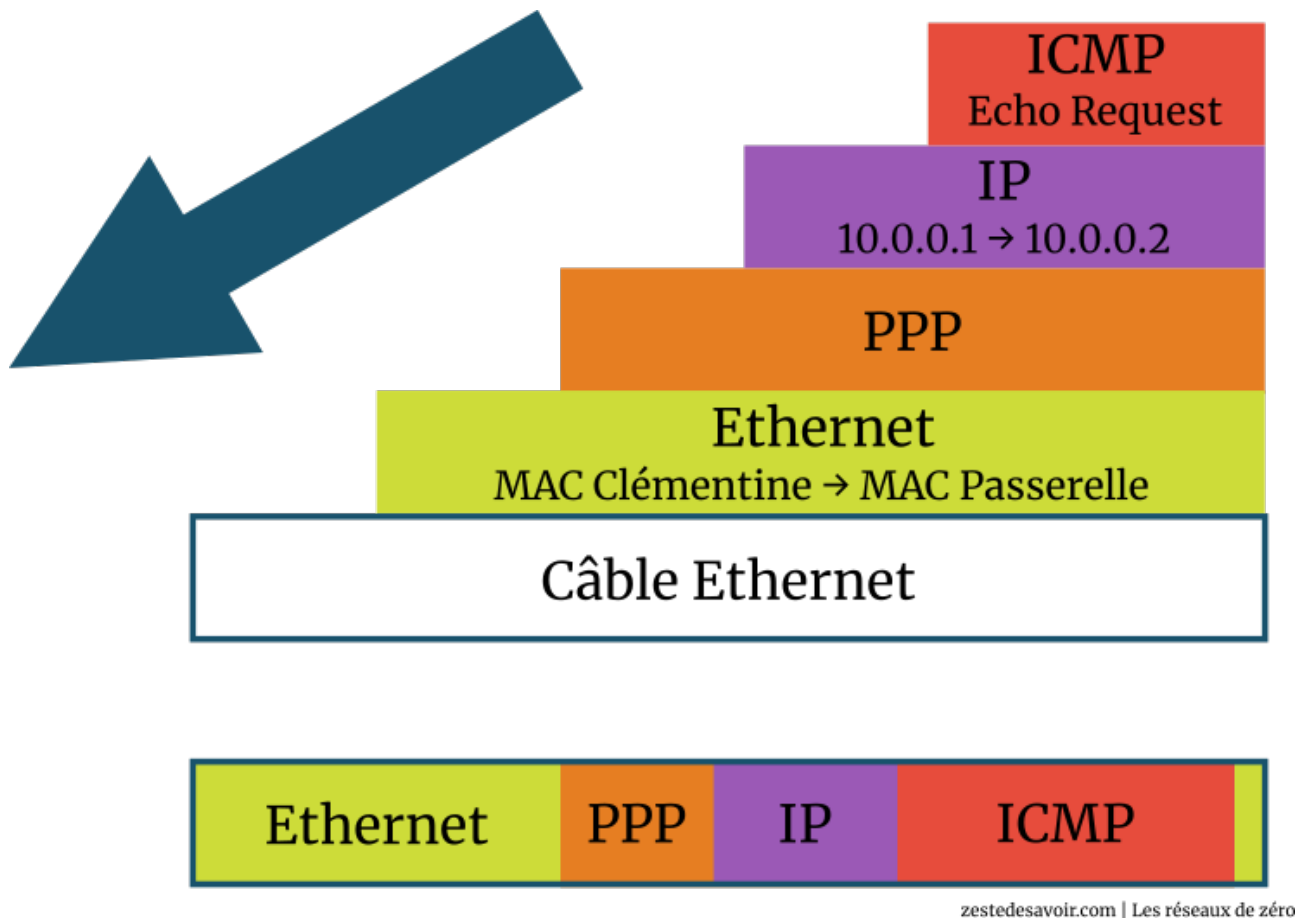


FIGURE V.3.2. – PPP encapsulé dans Ethernet (CC BY)

En pratique, quelle différence ? Dans le cas présent, ça ne change pas grand-chose. C'est surtout pour illustrer le rapport entre les interfaces. Là où ça se corse, c'est quand on veut faire de l'encapsulation au-delà du réseau strictement local.

### V.3.3. Le bout du tunnel

Dans le cas précédent, nous avons vu qu'un programme ne peut avoir conscience, au mieux, que de l'interface réseau qu'il utilise. Il ne sait pas ce qu'il y a "en dessous". Alors, si l'interface utilisée fait un traitement particulier des données qui transitent par elles, personne ne se rendra compte de rien tant qu'elles sont restituées correctement à l'autre bout. Là où c'est particulièrement intéressant, c'est si on crée une connexion point-à-point dont l'interface chiffre toute communication qui passe dessus. C'est possible avec le protocole IPsec.

i

IPsec est en réalité un agglomérat de protocoles qui permet de chiffrer l'intégralité d'un paquet IP et de ses données utiles. C'est un système assez complexe qui mériterait à lui seul plusieurs chapitres. Si vous vous sentez d'attaque, vous pouvez consulter [ce papier de FrameIP](#) .

### V.3.3.1. Au sec dans le tunnel

Reprenons notre exemple précédent et considérons maintenant qu'IPsec est appliqué à tout paquet qui passe par l'interface PPP. Voici ce qui se passe.

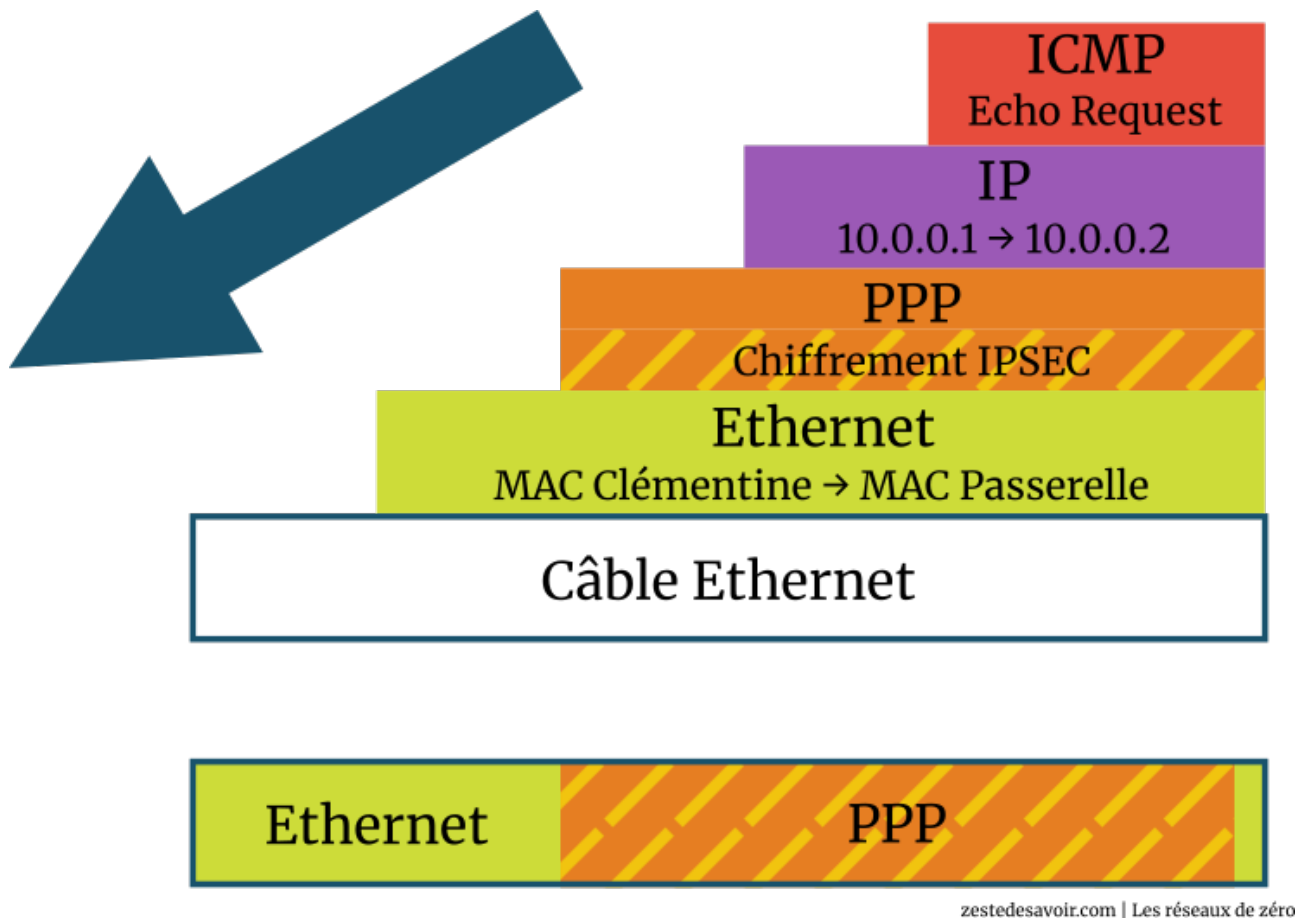


FIGURE V.3.3. – IPsec chiffre tout le contenu d'un paquet encapsulé dans PPP (CC BY)

On a formé ce qu'on appelle un **tunnel**. C'est une sorte de tuyau opaque où on peut voir ce qu'il y rentre et ce qui en sort, mais pas ce qui passe dedans. L'interface PPP de Clémentine représente un bout du tunnel, l'interface PPP de Mandarine représente l'autre extrémité. Entre les deux, tout est chiffré, pas moyen de savoir ce qui se trame dedans !

i

Il y a deux possibilités au niveau des adresses IP qui seront visibles. Soit on conserve les source et destination d'origine, c'est moins confidentiel car elles ne sont alors pas chiffrées. Soit on les remplace par la source et la destination des entités qui traitent le chiffrement et le déchiffrement. Dans ce dernier cas, c'est l'intégralité du paquet d'origine, en-tête inclus, qui est encapsulé dans un nouveau paquet IPsec. Dans notre exemple, ça revient au même.

C'est pas mal, car un tel modèle empêche une éventuelle interception sur un réseau Ethernet. Les switches indiscrets peuvent aller se rhabiller ! 🍌 Mais que diriez-vous de pousser le concept encore plus loin et l'étendre entre hôtes distants, par exemple sur Internet ?

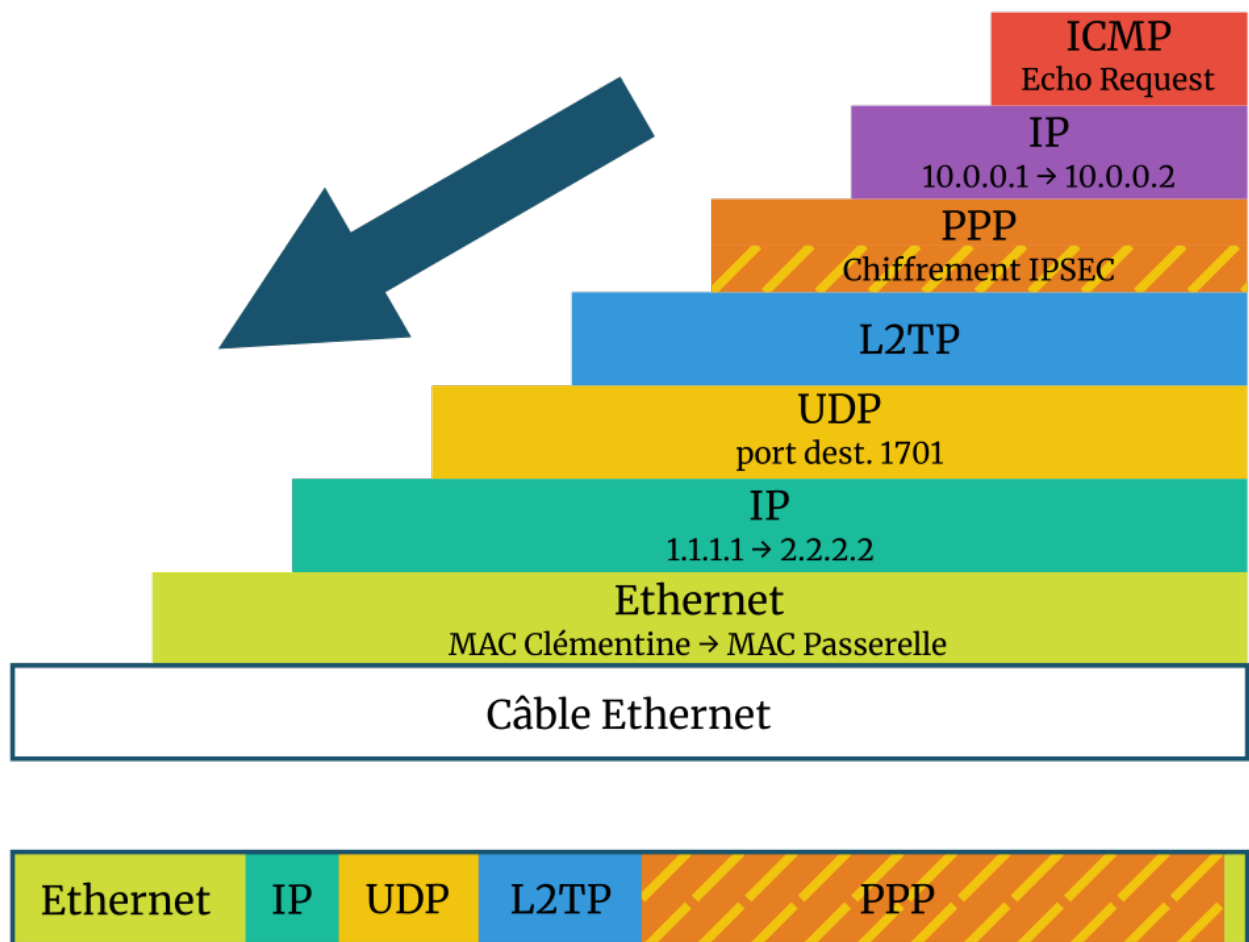
?

Pas possible avec PPP, c'est un protocole de niveau 2, ça reste restreint au réseau local !

On a bien encapsulé PPP dans Ethernet, alors pourquoi pas dans un protocole d'un autre niveau ? 🍊 L'idée peut paraître bizarre, mais il existe une solution : **L2TP**.

### V.3.3.2. Vers Internet et au-delà

L2TP, pour Layer 2 Tunneling Protocol, ou protocole de tunnel de couche 2. Tout est dit, ou presque. Ce protocole permet d'encapsuler PPP dans UDP. On peut faire transiter sans souci des paquets UDP sur Internet, alors puisqu'on peut encapsuler PPP dans UDP grâce à L2TP, eh bien mélangeons tout cela ! Pour couronner le tout, on va maintenant séparer physiquement Clémentine et Mandarine à deux endroits d'Internet. La première aura l'adresse 1.1.1.1, la seconde, 2.2.2.2.



zestedesavoir.com | Les réseaux de zéro

FIGURE V.3.4. – L2TP permet de transporter du PPP en dehors du réseau local (CC BY)

Ça commence à faire un beau mille-feuilles. Notez qu'au niveau IP inférieur, on reprend des adresses qui correspondent à des interfaces physiques. Cela devient nécessaire lorsqu'on veut fonctionner en dehors d'un réseau purement local : il faut que les routeurs sachent où orienter les paquets, et ils ne connaissent pas les adresses des interfaces PPP qu'on peut qualifier de virtuelles.

## V. On touche le fond !

Ce qui est vu dans le câble à partir du niveau IP est ce qui sera vu par tous les intermédiaires. Ainsi, si pour une raison quelconque, les paquets ICMP sont bloqués sur le réseau, ils passeront quand même entre Clémentine et Mandarine s'ils passent par le tunnel ! Eh oui, tout ce qu'on voit sur le réseau, ce sont des datagrammes UDP ! 🧙🏻‍♂️

Vous aurez peut-être remarqué que, depuis le début, on cherche à faire un ping entre 10.0.0.1 et 10.0.0.2. Ces hôtes sont théoriquement dans le même réseau IP. Et si vous vous souvenez bien, qui dit même plan d'adressage dit même réseau physique... Eh bien, avec L2TP, on a créé un **réseau privé virtuel**, un **VPN**. Les deux hôtes se voient comme voisins et comme faisant partie d'un même LAN, bien que ce ne soit physiquement pas le cas. Il existe d'autres moyens et d'autres protocoles pour établir des VPN : on peut citer [la solution OpenVPN](#) 🗨️ qui fonctionne différemment.

## Conclusion

Nous avons abordé beaucoup de notions complexes tout en restant en surface. Cela peut être frustrant et nous en avons bien conscience. Notre objectif est de vous donner de bonnes bases, pas de faire de vous des experts. Chacun des termes évoqués dans ce chapitre mériterait qu'on y passe des heures, qu'on y consacre un livre entier. C'est pourquoi nous préférons vous renvoyer vers d'autres ressources si vous souhaitez en savoir davantage et aller plus en profondeur.

Avant de quitter la couche 2 pour arriver vers le niveau physique, que diriez-vous d'un petit voyage dans le cœur des réseaux d'opérateur ? Embarquement immédiat pour le chapitre suivant !



## V.4. MPLS, un protocole qui en a le cœur net

### Introduction

Quand on commence à travailler pour de grandes entreprises, on découvre qu'elles disposent bien souvent de plusieurs sites géographiquement séparés et reliés par un VPN. Instinctivement, on se dit que les communications passent alors par Internet. C'est techniquement possible mais dans les faits, c'est rarement le cas. Pourquoi ? Par où les données circulent-elles ? Qu'est-ce que ces autres réseaux ont de plus ? C'est ce que nous allons voir dans ce chapitre.

#### V.4.1. Les réseaux d'opérateur privés

Si on évoque des noms d'opérateurs comme Orange, British Telecom, ou encore AT&T, il est fort probable que cela vous évoque l'accès à Internet. C'est vrai que ces sociétés proposent ce service, mais pas que. En plus des infrastructures d'interconnexion qui forment une partie d'Internet, ces entreprises disposent d'un autre réseau qui ne sert qu'à un nombre limité de clients, et principalement pour relier leurs différents sites entre eux. Plusieurs raisons justifient le besoin de disposer d'une infrastructure dédiée.

L'une d'entre elles est la sécurité, au sens large. Même si les communications peuvent être chiffrées, et qu'il n'est à priori pas grave qu'elles puissent être interceptées, on veut éviter qu'elles puissent être coupées par un tiers, quelle qu'en soit la raison. Lorsqu'un flux réseau passe par Internet, on ne contrôle pas par où les paquets circulent, on sait juste qu'ils suivent une route jusqu'à destination. Cette route peut emprunter des réseaux de plusieurs fournisseurs de service, passer par plusieurs pays, et changer d'un paquet à l'autre sans prévenir. Certaines communications critiques, par exemple liées à la sûreté des personnes (hôpitaux, police), nécessitent que l'on sache et que l'on contrôle où passent les flux.

Pour des raisons similaires, il peut être nécessaire d'avoir des garanties en termes de performances, comme le débit ou la latence. On appelle cela de la **qualité de service** (QoS, *Quality of Service*) et il n'est pas possible de proposer cela sur Internet, puisque comme évoqué, on ne contrôle pas l'infrastructure qui fait transiter la communication de bout en bout. Nous avons évoqué ce point précis dans le chapitre sur IP. Sur un réseau privé, on fait ce qu'on veut ! Il est donc possible de prioriser des flux par rapport à d'autres pour garantir que la visio du PDG dispose d'une bande passante suffisante pour ne pas saccader si tous les employés se mettent à regarder une série sur Netflix (pendant leur pause déjeuner, bien sûr).

En cas de panne, les opérateurs offrent des **garanties de temps de rétablissement** (GTR) à leurs clients professionnels. Cette forme d'assurance, qui augmente naturellement le prix de l'offre, impose à l'opérateur de rétablir le service en un temps spécifié (2 heures, 4 heures, etc.) sous peine de devoir payer des pénalités au client. Cette option peut être souscrite pour des accès à Internet, mais dans ce cas, cela ne garantit que... eh bien l'accès à Internet en général, et pas forcément au service auquel on souhaite accéder : si votre stockage en ligne est indisponible, cela n'est pas de la responsabilité de votre opérateur.

Il est aussi possible de recourir à un réseau d'opérateur privé pour disposer de services cloud

dédiés. Nous consacrerons un chapitre entier à ce sujet. Si vous utilisez Outlook ou Gmail pour tous les e-mails de votre entreprise, vous êtes dépendant de Microsoft ou Google et s'il y a un problème, c'est à ces entreprises que vous devrez vous adresser. En souscrivant à une offre qui inclut ces services, vous n'avez plus qu'un seul interlocuteur pour l'interconnexion de sites et les différents services dont vous avez besoin, ce qui est plus facile à gérer.



Bien entendu, tout cela a un prix. Là où un accès à Internet coûte quelques dizaines d'euros mensuels, passer par un réseau d'opérateur se chiffre en milliers d'euros par mois. On ne peut se permettre ces services que si le besoin est justifié !

Les cœurs de réseaux d'opérateur se doivent d'être performants pour offrir un niveau de service très élevé. Une technique pour cela tient en 4 lettres : MPLS.

## V.4.2. Mélange des genres

Un réseau d'opérateur est conséquent en termes d'infrastructures. Il peut même inclure des liaisons de différentes natures (fibres optiques, câbles Ethernet, ...) et reposer sur des protocoles de niveau 2 différents. Tout cela doit cohabiter harmonieusement pour que les paquets qui transitent soient routés jusqu'à bon port. IP fait bien cela, mais quand on y regarde de plus près, on se rend compte que cela nécessite à la fois du routage et de la commutation, et que le routage prend quand même du temps.



Ouh, routage, commutation... Quelle est la différence, déjà ? 🍌

La commutation consiste à renvoyer une trame (ou un paquet) sur une interface définie pour que l'information arrive à destination. Le routage... aussi. Mais le processus pour y parvenir est différent ! Lorsque l'on commute, on doit d'abord déterminer vers où renvoyer les données en fonction des informations du destinataire, comme l'adresse MAC de destination. Une fois que cela est fait, il n'y a plus à "réfléchir" : on renvoie toujours la trame sur la même interface pour ce même destinataire, tant que l'on ne reçoit pas de nouvelles informations qui invalideraient cette association (câble débranché, par exemple). Sur les réseaux Ethernet, les commutateurs apprennent et enregistrent ces associations dans une table de correspondance en analysant les adresses sources des trames qu'ils reçoivent sur chacun de leurs ports. Cela prend un petit peu de temps au début, mais le processus de commutation est ensuite très rapide, de l'ordre de la dizaine de microsecondes ( $10^{-5}$  s).

Le routage fonctionne différemment. Les routeurs disposent d'une table de routage qui peut évoluer sans prévenir si un protocole de routage dynamique est actif, ou bien si vous modifiez manuellement sa configuration de routage statique. Pour chaque paquet reçu, le routeur doit analyser sa table de routage et déterminer quelle sera l'interface de sortie en fonction de paramètres comme l'adresse IP de destination ou la métrique, qui "départage" plusieurs possibilités dans une table de routage. Cela nécessite des calculs qui prennent un peu de temps, de l'ordre d'un dixième de milliseconde ( $10^{-4}$  s) voire une milliseconde ( $10^{-3}$  s).

Ces temps paraissent ridiculement faibles, mais on parle tout de même d'un traitement 10 à 100 fois plus rapide avec la commutation par rapport au routage. Lorsqu'on souhaite proposer des temps de latence ultra faibles, cela a un impact. C'est là qu'intervient MPLS, pour *Multi*

## V. On touche le fond !

*Protocol Label Switching.* Ce protocole s'intercale entre les couches 2 et 3 et, dans une certaine mesure, se substitue au routage. Pour illustrer son fonctionnement, prenons le schéma suivant comme exemple. Au centre, la partie en rouge correspond à un réseau d'opérateur. Il compte de nombreux routeurs, ce qui implique un certain temps de routage d'un bout à l'autre. Ce cœur de réseau relie les deux sites d'un client vert d'une part, les deux sites d'un client bleu d'autre part.

i

Le terme **CE** signifie *Customer Edge* ("côte client") et est couramment employé pour désigner un routeur physiquement présent chez un client, pour le relier au réseau d'opérateur. De l'autre côté du lien se trouve un **PE**, pour *Provider Edge* ("côté fournisseur"), c'est-à-dire un routeur chez l'opérateur qui relie le client au cœur du réseau.

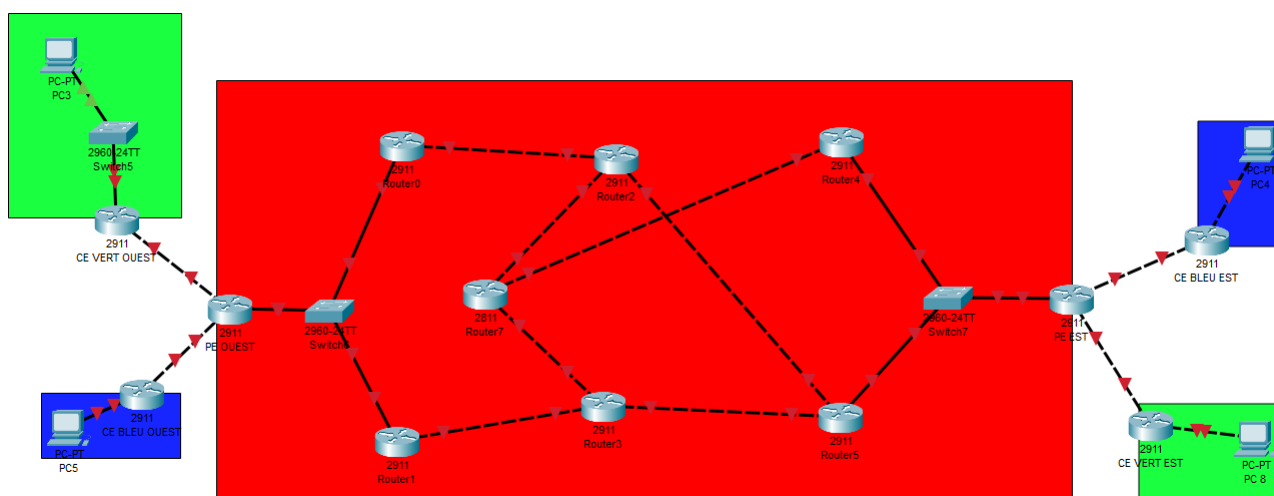


FIGURE V.4.1. – La partie centrale en rouge correspond au réseau d'opérateur et comprend de nombreux routeurs

Pour fonctionner correctement, le service MPLS est activé sur toutes les interfaces faisant partie de la zone cœur, en rouge. Cela exclut donc les interfaces des PE reliées aux CE. Les routeurs se constituent alors une base d'information que l'on appelle **LIB** (*Label Information Base*), grâce au protocole **LDP** (*Label Distribution Protocol*). Ce dernier définit la structure des messages que se transmettent les routeurs MPLS pour établir une LIB cohérente en fonction de **classes d'équivalence** (FEC, *Forwarding Equivalence Class*) qui correspondent à des règles de catégorisation pour déterminer quels flux doivent être rassemblés sous la même **étiquette**. En général, on se base sur les adresses des réseaux définis dans les tables de routage, mais on peut aussi définir des classes d'équivalence selon les protocoles transportés, les adresses source, etc. Une étiquette, ou *label*, fonctionne un peu comme une adresse MAC de destination : le paquet sera transmis sur une interface correspondant à l'étiquette, selon les informations de la LIB.

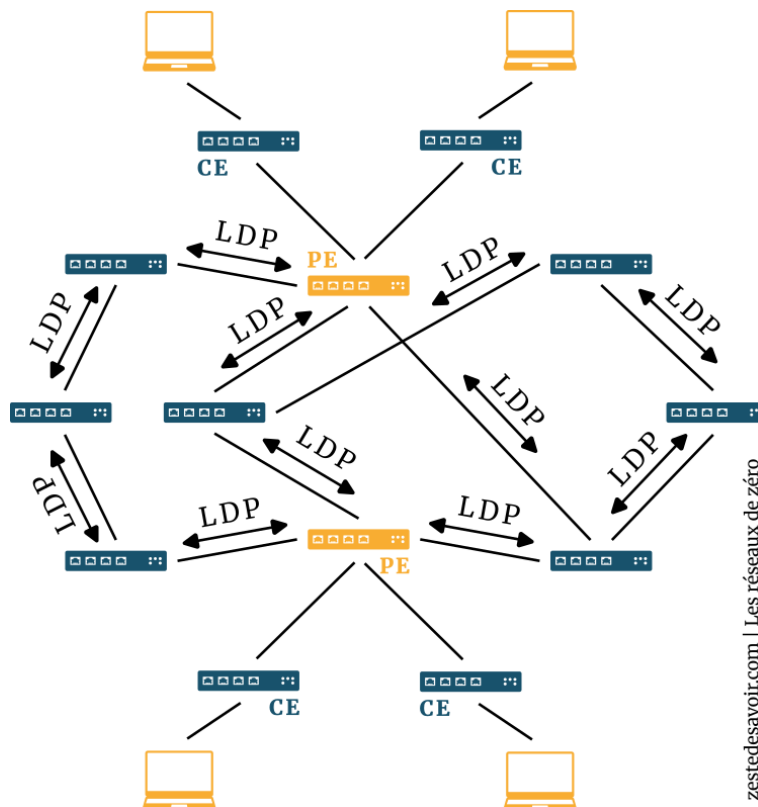


FIGURE V.4.2. – Les équipements échangent des informations grâce au protocole LDP dans un réseau MPLS (CC BY)

Une fois que les routeurs ont procédé aux échanges adéquats, les flux peuvent être commutés grâce à MPLS plutôt que routés de manière classique. Dans notre exemple, les routeurs PE correspondent à des **routeurs d'extrémité** (LER, *Label Edge Router*). Ils sont en charge d'introduire l'étiquette MPLS entre les couches 2 et 3, soit dans notre cas entre l'en-tête Ethernet et l'en-tête IP. Une fois le paquet arrivé à l'autre extrémité, l'autre LER retire cette étiquette et le routage habituel se poursuit en direction du CE. Dans le cœur du réseau, les autres routeurs agissent en LSR (*Label Switch Router*). Leur rôle est simplement de commuter les paquets selon l'étiquette. Plus besoin de recourir au long processus de routage, ils se contentent d'orienter le paquet sur l'interface qui correspond à l'étiquette dans la LIB.

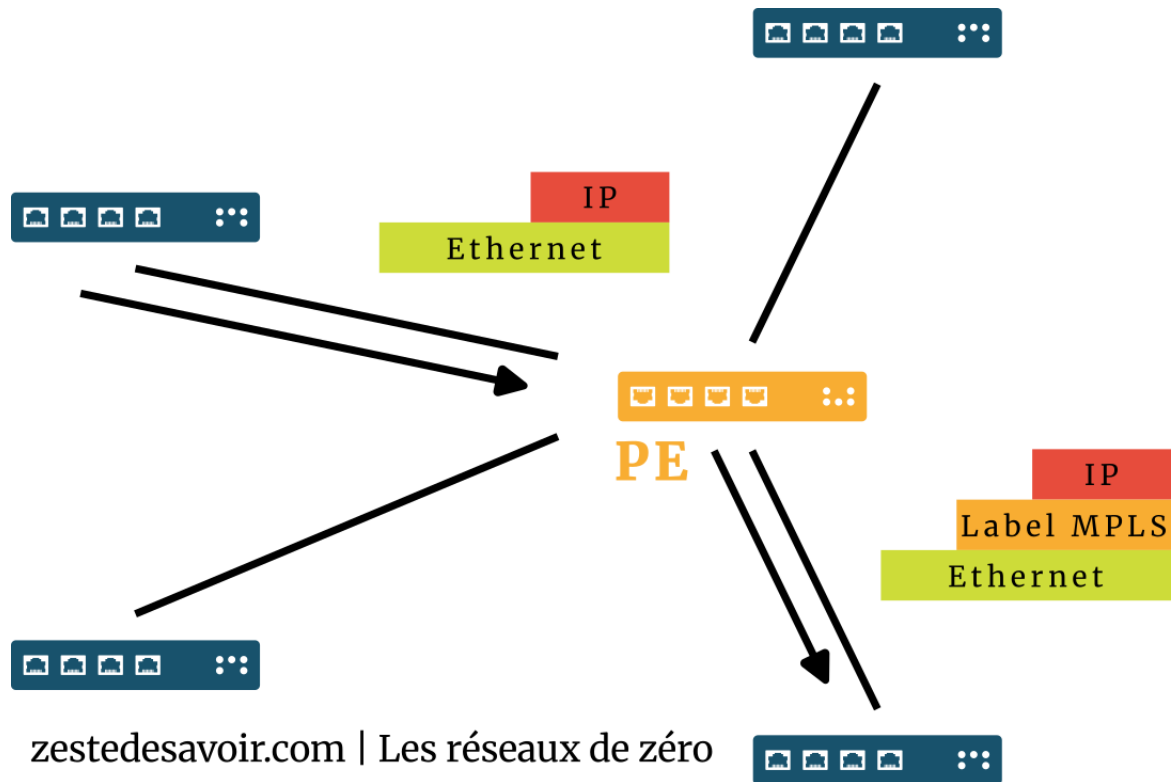


FIGURE V.4.3. – En arrivant dans un réseau MPLS, les paquets reçoivent une étiquette qui facilite leur orientation sur le réseau (CC BY)

Et voilà comment on peut augmenter les performances d'un cœur de réseau ! 🍊


## Conclusion

L'utilisation de MPLS sur les cœurs de réseaux opérateur fournissant des VPN est courant, à tel point qu'on dit parfois par abus de langage "le MPLS" pour désigner le réseau d'interconnexion en soi. Il présente l'avantage de fonctionner avec la plupart des protocoles de liaison de données, et donc sur la plupart des supports de transmission physiques. C'est justement ce point que nous allons aborder dans le prochain chapitre.

## V.5. Soyez au courant

### Introduction



Soyons honnêtes, nous ne sommes pas très calés sur le sujet évoqué et ce chapitre contient certainement beaucoup d'approximations. Tout retour permettant d'améliorer ce contenu est le bienvenu sur [le sujet du forum](#) .

Il fallait bien que ça arrive un jour : nous arrivons à la couche physique. Elle est très particulière, car ce n'est plus vraiment du réseau. On touche à une autre discipline qui est celle des télécommunications. Si on creuse encore plus, ça relève de l'électronique, et si on va toujours plus loin, de la physique.

Pour rester dans notre domaine des réseaux informatiques, nous n'introduirons que quelques notions élémentaires d'électronique. Cela devrait permettre à tous de suivre ce chapitre, qui s'axe autour de trois aspects des transmissions : l'accès au support physique, la synchronisation et la représentation des valeurs.

### V.5.1. Attention à la trame !

Tout au long de ce cours, nous avons mentionné différents types de matériel, de liaisons, de câbles, etc. Voyons maintenant ce qui passe au travers de ces éléments, et plus particulièrement celui que nous avons certainement le plus évoqué : le câble Ethernet.

#### V.5.1.1. La transmission dans les grandes lignes

Ce type de câble dispose de deux connecteurs, un à chaque extrémité. Le but de ces connecteurs est d'établir un contact physique avec les cartes réseaux. Entre eux, il y a 8 fils de cuivre. Ils sont enroulés entre eux, deux par deux, formant 4 **paires torsadées**. Deux de ces paires servent à la transmission des données jusqu'à 100 Mb/s : une pour émettre, une pour recevoir. Au-delà, les 4 paires sont utilisées. Les données sont physiquement représentées par des signaux électriques.

Mais ça, on en reçoit en permanence pour tout un tas de raisons. L'alimentation d'un ordinateur ou de n'importe quel appareil électrique génère un rayonnement électromagnétique, autrement dit, des ondes, qui sont elles-mêmes des signaux électriques. Un fil de cuivre a tendance à tellement bien conduire l'électricité qu'il absorbe toutes ces ondes qui l'entourent ! Ces parasites peuvent créer des **interférences**, c'est-à-dire des perturbations dans les signaux de transmission de données. Différentes solutions existent pour les éviter ou les gérer. Physiquement, il existe le **blindage**, qui consiste à envelopper les fils dans de l'aluminium, qui repousse les ondes. Mais il y a aussi un procédé logique que nous avons déjà vu : les sommes de contrôle ! Eh oui, en cas d'interférence qui viendrait altérer une donnée, la checksum peut nous avertir d'une erreur de transmission. 🍊

### V.5.1.2. Quels signaux ?

Pour donner une valeur logique à un signal électrique, on joue généralement sur sa tension. On peut décider arbitrairement qu'une tension de 5 volts correspond à un bit allumé, et une tension de 0 volt à un bit éteint. Seulement, un problème va vite s'imposer avec un tel système : dès qu'on veut transmettre deux 1 ou deux 0 consécutifs, on ne va pas savoir les repérer ! Un autre paramètre rentre alors en jeu : **l'horloge**.

Pour pouvoir savoir exactement quand il faut prendre en considération le signal électrique, on introduit la notion de **synchronisation**. Cela consiste à s'accorder entre les deux cartes réseaux pour se dire, par exemple, qu'une valeur est transmise chaque microseconde.





Nous disons bien une valeur, et non un bit. Comme on peut théoriquement fixer la correspondance qu'on veut entre tension et valeur logique, on pourrait très bien décider qu'un signal représente 2 bits : 0 volt pour 00, 5 volts pour 01, 10 volts pour 10, 15 volts pour 11. On parle alors de **bauds**. Dans ce dernier exemple, 1 baud vaut 2 bits. Mais dans le premier exemple, avec seulement deux valeurs, 1 baud vaut 1 bit.

Pour se synchroniser, les cartes réseaux doivent respecter la même norme. Elles savent donc dès le départ à quelle fréquence elles doivent regarder l'état du signal électrique qu'elles reçoivent. Cela peut être toutes les microsecondes, toutes les nanosecondes, ... Elles disposent d'une horloge qui doit être la plus précise possible. Malheureusement, il peut arriver qu'une perturbation quelconque vienne désynchroniser une horloge et qu'une information soit perdue ou mal interprétée. Nous verrons dans la section suivante comment des **codages** peuvent éviter cela.

### V.5.1.3. Et avec d'autres supports ?

Les principes évoqués ici sont applicables pour les supports à base de fils de cuivre. Ils peuvent être adaptés dans une certaine mesure à d'autres moyens de transmission, comme la fibre optique qui exploite des signaux lumineux, ou le Wi-Fi qui envoie ses ondes dans l'air sans support physique. Pour aller plus loin, voici quelques pistes qui vous orienteront vers des explications plus précises.

- [How Does an Optical Fiber Transmit Light ?](#)  par Craig Freudenrich (en anglais). Ce court document donne une première approche sur le fonctionnement des transmissions par fibre optique.
- [WiFi - Le Standard 802.11 - Couche physique et couche MAC](#) , par Michel Terré, Conservatoire National des Arts et Métiers. La couche physique du Wi-Fi y est expliquée à partir de la page 8. Cela demande quelques notions de base de télécommunications.

## V.5.2. Fais le baud

Nous avons vu qu'un signal électrique peut représenter une valeur en fonction de sa tension. La définition de cette équivalence, c'est ce qu'on appelle un **codage**. Il en existe un paquet ! Focalisons-nous sur 3 d'entre eux, pour pouvoir comprendre pourquoi il n'y en a pas de parfait.

### V.5.2.1. Le plus simple : NRZ

NRZ, pour *Non Return to Zero* (pas de retour à zéro), est probablement le plus simple des codages. On choisit une tension, par exemple 12 volts. Si on veut transmettre la valeur 1, on envoie un signal à +12 volts. Si on veut transmettre la valeur 0, on envoie un signal à -12 volts. On peut aussi inverser les valeurs, ça ne change rien. Le hic, c'est qu'à aucun moment, la tension ne revient à zéro, c'est même pour ça que ce codage porte ce nom... Bon courage pour s'y retrouver lors d'une suite de 0 ou de 1 ! L'horloge, d'un côté comme de l'autre, n'a pas le droit au moindre petit écart. La moindre perturbation peut corrompre la communication.



FIGURE V.5.1. – 10101100 : on peut encore s'y retrouver (CC BY)

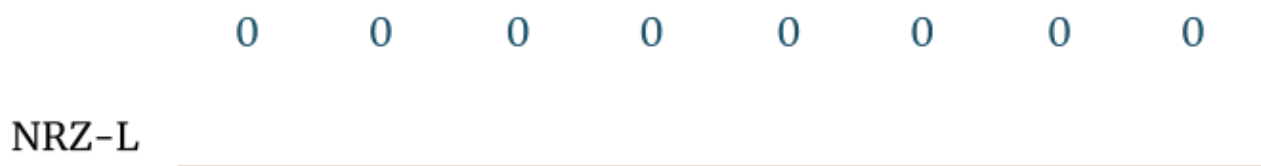


FIGURE V.5.2. – 00000000 : euh ça fait combien de zéros, là ? (CC BY)

Ce codage n'est pas adapté aux réseaux d'aujourd'hui. Il était utilisé à la fin du XX<sup>ème</sup> siècle dans les liaisons séries, qui sont en quelque sorte l'ancêtre des liaisons USB.

### V.5.2.2. MLT-3

MLT signifie Multi Level Transmit, soit transmission multi-niveaux. Le 3 indique que 3 niveaux sont utilisés. Avec ce codage, on prend une tension de référence, 12 volts pour ne pas changer. Il y a alors trois possibilités de signaux : +12 volts, 0 volt et -12 volts. On part de la tension négative. Pour transmettre la valeur 1, on fait varier le signal. Mais attention ! La variation ne peut avoir lieu que dans l'ordre suivant :

$-12\text{ V} > 0\text{ V} > +12\text{ V} > 0\text{ V} > -12\text{ V} > 0\text{ V} > +12\text{ V} > 0\text{ V} > \text{etc.}$

Et pour transmettre la valeur zéro... on ne change rien. Cela donne des variations comme sur l'image ci-dessous.

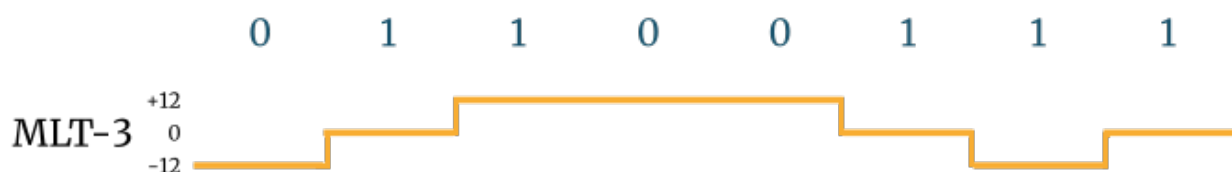


FIGURE V.5.3. – 01100111 : facile ! (CC BY)

## V. On touche le fond !

Ce codage présente l'avantage de bien gérer les suites de 1. La variation régulière peut permettre aux horloges de bien se calibrer et conserver la synchronisation. *A contrario*, pour une suite de 0... 🍊

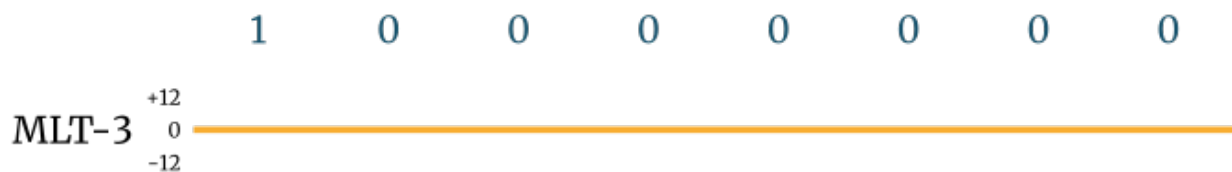


FIGURE V.5.4. – 10000000 : et zut... (CC BY)

On se retrouve avec le même problème que NRZ. On ne peut donc pas l'utiliser avec un protocole qui va générer de longues suites de 0... À moins d'utiliser une astuce qu'on garde pour la fin. 🍊

Ce codage est utilisé notamment dans la norme 100BASE-TX, qui définit les transmissions à 100 Mb/s en Ethernet.

### V.5.2.3. Manchester

Le codage Manchester présente une particularité : il embarque le signal d'horloge. Concrètement, il y a au moins une variation de la tension à la fréquence de la synchronisation. Attention, c'est plus complexe que les précédents.

Il y a deux états de tension possibles. Pour rester sur la même lignée, on va prendre pour exemple +12 volts et -12 volts. Il faut aussi prendre en compte la fréquence de l'horloge. Considérons que la période de synchronisation soit d'une microseconde. Lorsqu'on veut transmettre la valeur 1, il faut qu'au début de la période, la tension soit basse (-12 V), puis qu'elle passe haute (+12 V) à la mi-temps. Ainsi, si on avait transmis un autre 1 juste avant, la tension serait haute à la fin de la période précédente. Elle doit alors passer basse au début de la nouvelle période.



FIGURE V.5.5. – 111 en codage Manchester (CC BY)

Pour transmettre la valeur 0, c'est l'inverse. La tension doit être haute en début de période et passer basse à la mi-temps. Si on transmet un 0 puis un 1, la tension va rester basse à la fin de la première période et en début de seconde période.



FIGURE V.5.6. – 011 en codage Manchester (CC BY)



Historiquement, c'est dans l'autre sens. Le 0 allait de bas en haut et le 1 de haut en bas. Par la suite, et notamment pour Ethernet, cela a été inversé.

Comme la tension varie avec une fréquence au moins équivalente à l'horloge, on évite le risque de désynchronisation. Par contre, on transmet deux fois plus de signaux, ce qui réduit fatalement le débit du support.



FIGURE V.5.7. – 11111111 en codage Manchester (CC BY)

Ce codage est utilisé dans la norme 10BASE-T, servant aux transmissions à 10 Mb/s en Ethernet. Ça paraît ridicule mais c'était très courant à la fin du XX<sup>ème</sup> siècle ! Bon, il faut dire que Manchester n'est pas non plus tout jeune : il est utilisé depuis... les années 1940. 🍊

#### V.5.2.4. Et maintenant, une astuce

Nous avons soulevé un problème de certains codages qui est l'absence de variation en cas de suite de valeurs identiques. Pour le pallier, il existe une astuce qui consiste à substituer une suite par une autre. 🍊

L'exemple le plus simple est celui utilisé par l'entreprise Oregon Scientific pour ses stations météo. Comme décrit dans [ce papier](#) ↗ , il consiste à envoyer, après chaque bit, son complément... c'est-à-dire son contraire. Pour envoyer la valeur 1, on envoie 10, et inversement. Ainsi, la suite 00000000 devient 01010101010101 ! Ça prend deux fois plus de temps à transmettre mais ça fonctionne. 🍊

Plus économique en place, d'autres systèmes existent, comme le 4B5B. Le principe consiste à remplacer une suite de 4 bits par une autre suite de 5 bits avec le moins de valeurs identiques qui se suivent. Ainsi, on remplace 0000 par 11110, 0001 devient 01001, etc. Vous retrouverez [la liste complète ici](#) ↗ . Cela permet aussi d'identifier des erreurs de transmission, car certaines combinaisons sont impossibles. C'est moins lourd que le système précédent, car cela n'augmente le nombre de transmission que de 25%. Dans la même veine, il existe aussi [le 8B10B](#) ↗ .

Il existe bien d'autres codages mais aucun n'est parfait. Ceux que nous avons passé en revue doivent vous permettre de comprendre les principes de base. 🍊

## Conclusion

Il est loin, le temps où André demandait à Jean de transmettre un livre à Pierre ! Maintenant, vous en êtes à vous interroger sur la cadence des variations de tension dans un fil électrique ! 🍊  
Que de chemin parcouru... Que diriez-vous d'un récapitulatif global ? Dans le prochain chapitre, on remonte les couches une par une et on révise les principales notions étudiées tout au long du cours. 🍊

## V.6. Remontons à la surface

### Introduction

Depuis le début de ce cours, vous avez appréhendé beaucoup de notions. À partir d'une seule trame, nous allons remonter les couches du modèle OSI et résumer leur rôle. Pour cela, nous utiliserons un analyseur de trame, qui nous permettra de visualiser bit par bit ce qui passe par le support physique.

#### V.6.1. Contexte

L'expérience réalisée pour ce chapitre est simple. Depuis un PC, on a ouvert un navigateur et on est allé sur [zestedesavoir.com](http://zestedesavoir.com). Ce PC est relié à un routeur, lui-même connecté à Internet. En même temps, on a lancé une capture de trame avec le logiciel Wireshark. Ce programme permet de visualiser finement ce qui se passe au niveau des interfaces réseaux, à partir du niveau 2. [Vous pouvez apprendre à l'utiliser grâce à cette annexe.](#) ↗

La capture réalisée permet de visualiser dans le détail chaque valeur de chaque champ de chaque protocole. Nous allons partir de la couche liaison de données et remonter jusqu'à la couche application. D'abord, nous prendrons le point de vue de la **carte réseau** pour les niveaux 2 et 3, puis celui du **processus** pour les niveaux 4 et 5, et enfin celui de l'**application** pour les niveaux 6 et 7.

```
> Frame 174: 1423 bytes on wire (11384 bits), 1423 bytes captured (11384 bits) on interface 0
> Ethernet II, Src: Sagemcom_7[redacted]b (b0:b2:8f:7[redacted]b), Dst: IntelCor_[redacted]a (30:24:32:[redacted]a)
> Internet Protocol Version 6, Src: 2a01:111:2010:8::ff20, Dst: 2a01:cb14:457:ff00:f15d:4f88:b57:8f2b
> Transmission Control Protocol, Src Port: 443, Dst Port: 53016, Seq: 16276, Ack: 5862, Len: 1349
> [8 Reassembled TCP Segments (11429 bytes): #167(1440), #168(1440), #169(1440), #170(1440), #171(1440), #172(1440), #173(1440), #174(1349)]
> Transport Layer Security
```

FIGURE V.6.1. – Représentation de la trame



Les adresses MAC ont été partiellement masquées pour des raisons de confidentialité.

Voici comment se représente la trame que nous allons décortiquer. Si vous avez l'œil, vous aurez peut-être déjà repéré qu'il s'agit d'une réponse du serveur interrogé qui nous parvient. 🍊

#### V.6.2. Au niveau de la carte réseau

Rentrons dans le détail de la couche 2.

```
▼ Ethernet II, Src: Sagemcom_7[redacted]b (b0:b2:8f:7[redacted]b), Dst: IntelCor_[redacted]a (30:24:32:[redacted]a)
  > Destination: IntelCor_[redacted]a (30:24:32:[redacted]a)
  > Source: Sagemcom_7[redacted]b (b0:b2:8f:7[redacted]b)
  Type: IPv6 (0x86dd)
```

FIGURE V.6.2. – Détail du protocole de niveau 2

Il s'agit du protocole Ethernet. On y retrouve les adresses MAC et l'EtherType, noté simplement "Type". Les adresses MAC représentent des identifiants physiques et uniques des hôtes, ou plus exactement, de leur carte réseau. La destination est déterminée par la source en fonction des besoins du protocole de niveau supérieur. L'EtherType nous indique qu'il s'agit d'IPv6. L'adresse MAC de destination doit donc correspondre à celle de l'hôte de destination niveau IP, ou, s'il n'est pas dans le réseau local, à celle d'une passerelle adéquate.

Avouez qu'Ethernet, c'est plutôt simple ! On n'a pas de VLAN dans notre cas, ce qui facilite encore plus les choses. On n'a pas non plus de liaison point-à-point à transporter. Si vous éprouvez le besoin de réviser aussi ces sujets, n'hésitez pas à relire les chapitres correspondants.



Passons maintenant à la couche réseau et son paquet IPv6.

```
▼ Internet Protocol Version 6, Src: 2a01:111:2010:8::ff20, Dst: 2a01:cb14:457:ff00:f15d:4f88:b57:8f2b
  0110 .... = Version: 6
  ▼ .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 00.. .... = Differentiated Services Codepoint: Default (0)
    .... ..00 .... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    .... .... 0000 0000 0000 0000 = Flow Label: 0x000000
  Payload Length: 1369
  Next Header: TCP (6)
  Hop Limit: 111
  Source: 2a01:111:2010:8::ff20
  Destination: 2a01:cb14:457:ff00:f15d:4f88:b57:8f2b
```

FIGURE V.6.3. – Détail du protocole de niveau 3

IP permet la transmission de paquets sur le réseau d'un bout à l'autre. Pour cela, il embarque un certain nombre de fonctionnalités qui se déclenchent selon divers paramètres. Le premier champ précise qu'il s'agit de la version 6. C'est important à savoir pour les routeurs : sans ça, ils ne sauraient pas forcément détacher et interpréter correctement les valeurs qui suivent !

On retrouve ensuite le DSCP. Il permet de donner une information sur le traitement à apporter au paquet, dans le cadre de la qualité de service. Ce système ne peut être utilisé que sur un réseau DiffServ, qui gère la priorité des paquets en fonction de divers paramètres comme le contenu transmis. Comme on n'est pas dans ce cas ici, ce champ vaut zéro.

L'ECN est un drapeau double permettant d'avertir les hôtes d'une congestion sur le réseau. Dans notre exemple, il n'est pas supporté. Le *flow label*, qui permet d'étiqueter des paquets appartenant à un même flux, n'est pas utilisé non plus. De manière générale, ces trois éléments ne sont pas utilisés sur Internet. Ils sont réservés aux réseaux d'entreprise.

La *payload length*, taille des données utiles, nous indique que 1369 octets sont encapsulés dans ce paquet. Le *next header* nous informe qu'ils correspondent à un segment TCP. IPv6 supporte un système d'extensions qui peuvent se superposer. Ainsi, un *next header* peut aussi indiquer qu'il faut s'attendre ensuite à un complément d'en-tête, par exemple une information de fragmentation. Un système similaire d'options existe avec IPv4 mais n'est presque pas utilisé.

Le *hop limit* qui suit, qu'on peut aussi appeler *Time To Live*, indique le nombre de sauts que le paquet pouvait encore effectuer sur le réseau. Un saut correspond à la traversée d'un routeur. Pour éviter qu'un paquet ne se retrouve coincé dans une boucle à cause d'une erreur, le nombre maximum de traversées est limité. Avec 111 sauts restants, il y avait de la marge. 🍊

Pour finir, on retrouve les adresses source et destination. Avec IPv6, elles sont codées sur 128 bits, soit 16 octets. Pour faciliter la lecture, on les écrit en hexadécimal par blocs de 4 chiffres,

## V. On touche le fond !

en retirant les zéros inutiles. S'il y a beaucoup de zéros consécutifs, on peut les compresser en écrivant juste « :: ».

?

On a dit au départ qu'on a juste tapé `zestedesavoir.com` dans un navigateur, alors comment on se retrouve avec ces adresses précises dans le paquet ? 🍊

Ça, c'est au programme de la prochaine partie du cours. Patience ! 🍊

### V.6.3. Au niveau du processus

Maintenant, prenons le point de vue processus. Pour les besoins de ce chapitre, on sépare processus et application, même si les deux correspondent au même logiciel. On considère que le processus, c'est la partie du navigateur qui gère les connexions et le transport (niveaux 4 et 5), tandis que l'application, c'est ce qui gère le niveau applicatif et permet l'affichage de pages web (niveaux 6 et 7). La partie processus pourrait être la même dans plein de programmes différents, tandis que l'application est spécifique au programme.

On en arrive à la couche transport. Si vous avez décroché lors des chapitres correspondants, c'est l'occasion de vous rattraper ! 🍊

```
▼ Transmission Control Protocol, Src Port: 443, Dst Port: 53016, Seq: 16276, Ack: 5862, Len: 1349
  Source Port: 443
  Destination Port: 53016
  [Stream index: 9]
  [TCP Segment Len: 1349]
  Sequence number: 16276      (relative sequence number)
  [Next sequence number: 17625      (relative sequence number)]
  Acknowledgment number: 5862      (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window size value: 64800
  [Calculated window size: 64800]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x9c13 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
  TCP payload (1349 bytes)
  TCP segment data (1349 bytes)
```

FIGURE V.6.4. – Détail du protocole de niveau 4

La couche transport permet d'établir une communication de bout en bout au niveau des processus. Pour identifier ces derniers sur un même système, on a recours à des **ports**, qui sont en quelque sorte des portes d'entrée. Ils sont numérotés de 0 à 65.535. Ceux entre 0 et 1023 sont dits "connus" et ont une application bien définie. Entre 1024 et 49.151, les ports sont dits "enregistrés". Certaines applications les utilisent de façon récurrente, mais leur utilisation est assez peu encadrée. Enfin, après 49.152, c'est libre. Cette plage est souvent utilisée par les clients pour des ouvertures de connexion qui n'ont pas besoin d'un numéro fixe et défini.

## V. On touche le fond !

Dans notre exemple, on voit que le port source est 443, et le port destination, 53.016. 443 est un port connu qui correspond à un serveur web, et plus précisément au protocole HTTPS. La destination est un port libre. On constate donc que cette transmission est en provenance d'un serveur web et à destination d'un client. C'est donc une réponse à une requête précédemment envoyée.

TCP est orienté connexion. Il utilise un système de **numéros de séquence**, qui lui permet de savoir où on en est dans le transport des données. Pour peu que la quantité de données à transporter soit importante, elles seront découpées en plusieurs segments et TCP permet de s'assurer qu'ils arrivent tous bien à destination. Ici, le numéro de séquence relatif est 16.276, ce qui signifie que ce qui vient fait suite au 16.276ème octet du flux. En même temps, on indique un **numéro d'acquittement** de 5.862. Cela précise que, dans l'autre sens, on attend de notre interlocuteur un 5.862ème octet pour sa prochaine transmission. Si l'autre hôte avait déjà transmis 6.000 octets, cela lui permettra de se rendre compte qu'un segment n'a pas été reçu et ainsi de le renvoyer.

i

Les valeurs exprimées ici sont relatives, pour que ce soit plus lisible. Une séquence peut être initialisée avec une valeur arbitraire pour des raisons de confidentialité. Dans notre exemple, le numéro de séquence réel est 479.815.025. C'est tout de suite moins parlant !

Un mot sur la **longueur de l'en-tête** : 5 blocs de 4 octets, ça fait 20 octets. Les données de niveau supérieur se trouveront donc après ces 20 octets d'en-tête. Ce champ peut servir si le segment présente des options, ce qui n'est pas le cas ici.

Les **drapeaux** sont plus intéressants. On a ici les flags PSH et ACK levés. PSH indique au système que la transmission doit partir tout de suite, sans attendre de remplir davantage une mémoire tampon. ACK précise à l'interlocuteur qu'il doit faire attention au numéro d'acquittement juste avant, car l'émetteur souhaite accuser réception de données.

Le champ qui suit est la **fenêtre**. L'émetteur indique qu'il peut recevoir jusqu'à 64.800 octets sans acquittement. L'hôte de destination est invité à patienter avant son prochain envoi s'il n'a pas reçu d'accusé de réception après 64.800 octets envoyés. Cela permet de réguler la vitesse du flux, d'avoir des envois plus réguliers et d'éviter d'être surchargé.

La **somme de contrôle** vérifie l'intégrité des données transmises. Le destinataire va la recalculer selon le même mode opératoire que l'émetteur. S'il ne tombe pas sur la même valeur, il considérera qu'il y a un problème et n'acquittera pas le segment.

Le dernier champ n'est presque jamais utilisé. Il sert à préciser quelle partie du segment doit être traitée urgemment. Il n'est lu que si le drapeau "URG" est levé. Son utilisation est anecdotique.

Au début du chapitre, une ligne a peut-être attiré votre attention dans la capture.

> [8 Reassembled TCP Segments (11429 bytes): #167(1440), #168(1440), #169(1440), #170(1440), #171(1440), #172(1440), #173(1440), #174(1349)]

FIGURE V.6.5. – Réassemblage de segments TCP

8 segments ont été réassemblés pour pouvoir lire le protocole de niveau supérieur. Il n'était pas possible de transmettre les 11.429 octets d'un coup, alors ils ont été segmentés pour être transportés. Le processus les a ensuite réunis pour pouvoir reconstruire les données d'origine.

En quoi consistent-elles, justement ? Eh bien, il s'agit de TLS, un protocole de chiffrement. Vous vous souvenez qu'à l'origine, on étudie la transmission d'une page web de Zeste de Savoir ? Pour cela, on utilise HTTPS, qui est la combinaison de HTTP, le protocole du web, encapsulé dans TLS, qui est en charge du chiffrement.

## V. On touche le fond !

Comme nous sommes dans un chapitre de révision, on vous épargne le fonctionnement de TLS. Au moment de la capture de la trame, les échanges pour définir les conditions du chiffrement ont déjà eu lieu. Tout ce qu'on voit, c'est ceci, et ça se passe d'explications.

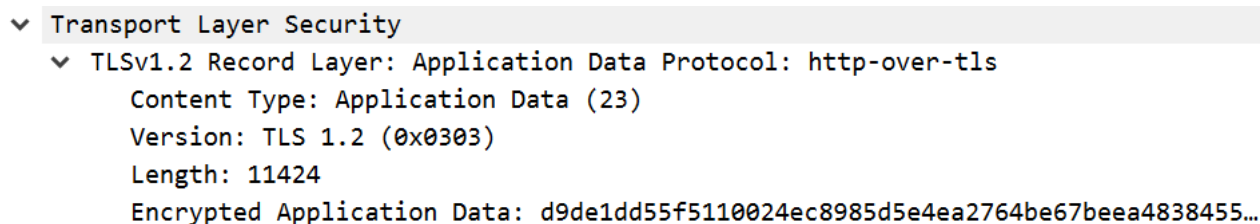


FIGURE V.6.6. – Détail du protocole de niveau 5

Le fait d'avoir écrit "niveau 5" va nous attirer des foudres. 🍊 Personne n'est d'accord pour associer TLS à une couche en particulier. Certains considèrent que cela relève du transport, d'autres, de la présentation ou encore de la session. Vous le verrez parfois classé dans les niveaux 4, 5 ou 6, voire à cheval sur plusieurs niveaux, ou pire : entre deux niveaux.

En revanche, on ne risque pas d'être remis en cause pour le niveau applicatif. 🍊

### V.6.4. Au niveau de l'application

On va avoir un souci avec notre analyseur de trame : à partir de maintenant, tout est chiffré ! Pour pouvoir visualiser ce qui est transmis au dessus de TLS, on a eu recours à un [proxy](#) 🍊. Ce service, qui sert de relai, sera étudié dans la partie suivante. 🍊  
Voici ce qui est vu par notre application.

1	HTTP/1.1 200 OK
2	Server: nginx
3	Date: Wed, 20 Mar 2019 22:45:08 GMT
4	Content-Type: text/html; charset=utf-8
5	Connection: close
6	Vary: Accept-Encoding
7	Vary: Cookie
8	Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
9	X-Xss-Protection: 1
10	X-Content-Type-Options: nosniff
11	X-Frame-Options: SAMEORIGIN
12	P3P: CP="ALL DSP COR PSAa PSDa OUR NOR ONL UNI COM NAV"
13	Strict-Transport-Security: max-age=15768000
14	X-Clacks-Overhead: GNU Terry Pratchett
15	Content-Length: 87430
16	
17	
18	
19	
20	
21	

## V. On touche le fond !

22	
23	
24	
25	
26	<!DOCTYPE html>

Ce serait beaucoup trop long de retranscrire l'intégralité de la réponse HTTP. Nous nous arrêtons à la première ligne après en-tête, c'est suffisant.

La couche 6 correspond à la présentation. Elle n'englobe pas de protocoles comme pour les autres niveaux. Elle désigne la façon dont l'information est représentée. Dans le cas de HTTP, l'en-tête, qui correspond au bloc ci-dessus à l'exception de la dernière ligne, est toujours codé en ASCII, un encodage simple faisant correspondre une valeur numérique sur un octet à un caractère. Ensuite, dans l'en-tête, on peut définir qu'on va faire autrement pour le contenu. Regardez cette ligne :

1	Content-Type: text/html; charset=utf-8
---	--

La fin indique que la page web qui suit sera encodée en UTF-8, un autre système de représentation de caractères. On aurait aussi pu avoir un en-tête indiquant que le contenu était compressé, ce qui n'est pas le cas ici. Cela relève aussi de la couche 6.

Finalement, on remonte à la surface avec le protocole de niveau 7, HTTP. Le contenu que nous avons là est une réponse d'un serveur web. Elle se lit facilement puisqu'elle est déjà bien présentée grâce à la couche 6. 🍊 On peut distinguer l'en-tête, que nous avons représenté en intégralité, et le contenu, dont nous n'avons représenté que la première ligne. Si vous voulez le reste, c'est simple : allez sur la page d'accueil de Zeste de Savoir et affichez le code source. 🍊

## Conclusion

Et maintenant ? Nous n'allons pas nous quitter comme ça. 🍊 Des questions ont été soulevées dans ce chapitre. L'objet de la partie suivante sera de revenir dessus en détail.

# Conclusion

Vous avez maintenant de solides connaissances sur le fonctionnement des principaux protocoles des réseaux, et sur la façon dont ils sont complémentaires. Dans la partie suivante, nous allons voir comment certains d'entre eux sont exploités pour rendre des services indispensables au quotidien.

# Sixième partie

## Reprenons du service

# Introduction

Vous disposez désormais de bonnes connaissances fondamentales sur les réseaux. Il est maintenant temps de visualiser comment tous ces principes sont mis en œuvre au quotidien pour que connecter son smartphone à Internet ne nécessite pas un doctorat en télécommunications (ou pire : de devoir lire intégralement *Les Réseaux de Zéro* 🍊 ).

## VI.1. DNS arrive à point nommé

### Introduction

S'il y a bien un service d'importance vitale sur le réseau, c'est le **système de noms de domaine** (DNS, *Domain Name System*). C'est grâce à lui que nous disposons d'adresses faciles à retenir, comme `zestedesavoir.com`, et que nous n'avons pas à retenir d'adresses IP. En effet, le concept du DNS est d'attribuer des noms à des adresses, pour que ce soit plus parlant aux utilisateurs. Nous commencerons ce chapitre par une présentation globale de ce système. Nous aborderons ensuite le principe de récursivité et la répartition des serveurs, puis nous nous pencherons sur les différents types de requêtes. Enfin, nous verrons quelles sont les évolutions actuelles du DNS qui tentent de pallier ses défauts.



DNS est un système devenu très complexe qui peut faire l'objet de livres entiers pour le maîtriser. Ce chapitre pose seulement les bases permettant de comprendre les fondamentaux du système. Rendez-vous dans la conclusion pour des pistes permettant d'approfondir le sujet !

#### VI.1.1. En bref

Si on veut résumer de manière succincte ce qu'est DNS, on peut dire que c'est un système permettant d'attribuer des noms à des adresses IP. Faisons tout de suite un test, ouvrez une console et tapez la commande suivante :

- Pour Windows : `nslookup zestedesavoir.com`
- Pour Linux et Mac OS : `dig zestedesavoir.com`

On obtient un résultat de ce genre sous Windows :

```
1 Serveur :      UnKnown
2 Address:  192.168.1.1
3
4 Réponse ne faisant pas autorité :
5 Nom :      zestedesavoir.com
6 Addresses: 2001:4b98:dc0:41:216:3eff:febc:7e10
7            92.243.7.44
```

Sous Linux :

## VI. Reprenons du service

```
1 ; <<>> DiG 9.10.3-P4-Debian <<>> zestedesavoir.com
2 ;; global options: +cmd
3 ;; Got answer:
4 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34910
5 ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL:
6     1
7 ;; OPT PSEUDOSECTION:
8 ; EDNS: version: 0, flags:; udp: 4096
9 ;; QUESTION SECTION:
10 ;zestedesavoir.com.                IN      A
11
12 ;; ANSWER SECTION:
13 zestedesavoir.com.                300     IN      A      92.243.7.44
14
15 ;; Query time: 89 msec
16 ;; SERVER: 192.168.1.1#53(192.168.1.1)
17 [...]
```

Ce retour nous donne plusieurs informations. Le plus important, ce sont les adresses associées au nom de domaine `zestedesavoir.com`, qui sont `92.243.7.44` et `2001:4b98:dc0:41:216:3eff:febc:7e10`. Par défaut, `dig` ne demande que l'adresse IPv4, c'est pourquoi nous ne voyons pas l'adresse IPv6 ici.



Un **nom de domaine** désigne un nom utilisé par DNS auquel on peut associer une ou plusieurs adresses IP. Il est composé de plusieurs parties séparées par des points.

Grâce à cette information, quand vous tapez `zestedesavoir.com` dans votre navigateur, il sait qu'il doit contacter une des adresses IP renvoyées.

L'autre information importante, c'est le serveur contacté, ici `192.168.1.1`. Le client (ici, `nslookup` ou `dig`) s'est adressé à ce serveur au moyen d'une requête DNS, envoyée en UDP sur le port 53. La réponse se fait par le même moyen. Nous verrons en fin de chapitre que ce fonctionnement pose des problèmes de sécurité. Avant cela, voyons pourquoi c'est ce serveur en particulier qui a été contacté et pas un autre.

### VI.1.2. Un arbre à l'envers

Dans la majorité des cas, le serveur DNS que notre ordinateur contacte est défini par le réseau. Notre système reçoit généralement cette information au moyen du protocole DHCP, qui fera l'objet du prochain chapitre. Ce serveur DNS fait souvent partie du réseau local, comme dans l'exemple précédent. Et notre pauvre serveur DNS local, eh bien, il ne peut pas connaître tous les noms de domaine du monde. 🍏

### VI.1.2.1. Passe à ton voisin

Quand il ne sait pas répondre, un serveur DNS transmet la requête à un autre, qui fait office de référent. Un cas classique : sur votre connexion Internet privée, votre routeur fait probablement office de serveur DNS. S'il ne sait pas répondre à une requête, il la transmet à un serveur DNS de votre opérateur, qui sera plus à même de connaître la réponse.

?

Pourquoi on ne contacte pas directement le serveur de l'opérateur, alors ?

Déjà, c'est plus rapide de contacter son voisin qu'un serveur on ne sait où. En plus, cela permet d'économiser des ressources. Le serveur DNS local apprend et enregistre les requêtes et les réponses associées et peut y répondre directement quand elles sont à nouveau demandées. Ces enregistrements sont valables pour une période de temps donnée : c'est le *Time To Live* (TTL). La durée de validité peut être d'une heure, un jour, ... On appelle cela un **cache**.

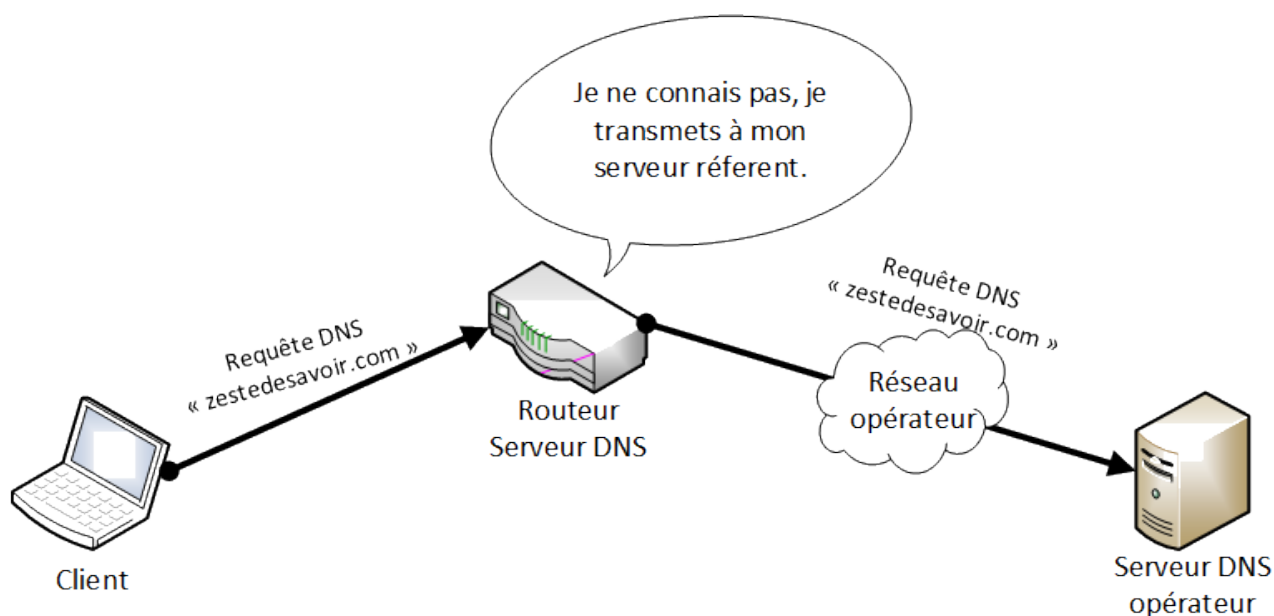


FIGURE VI.1.1. – Transmission d'une requête DNS

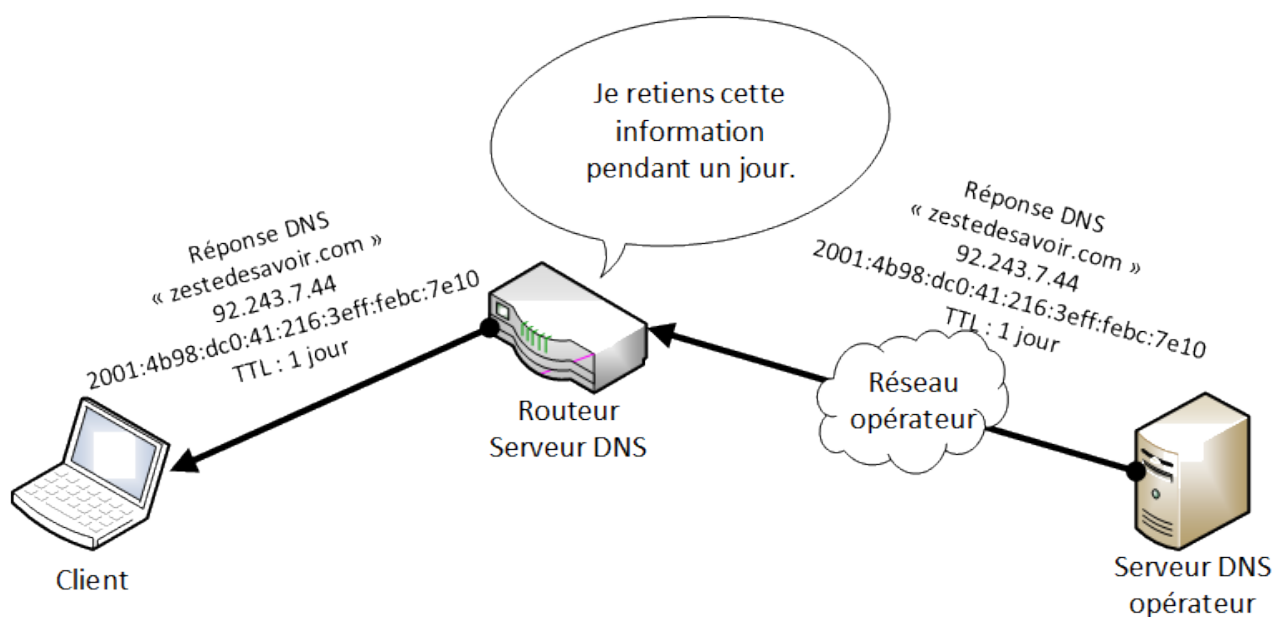


FIGURE VI.1.2. – Transmission d’une réponse DNS



Et le serveur de l’opérateur, il connaît forcément la réponse ?

Eh bien non, c’est pareil : quand il ne connaît pas, il demande et garde en mémoire la réponse. Il peut demander à d’autres serveurs qui sont ses propres référents, et ainsi de suite. Le dernier recours, ce sont les **serveurs racines**, ou *root servers*.

### VI.1.2.2. Par la racine

Il existe dans le monde plusieurs serveurs DNS un peu particuliers. On s’adresse à eux quand on est vraiment désespéré on n’a aucun autre serveur à contacter. Leurs adresses sont fixes et ils ne font que renvoyer vers les serveurs d’autorité du **domaine de premier niveau** (TLD, *Top Level Domain*). Cela correspond à la dernière partie du nom de domaine : .com, .net, .fr, ...



Un serveur DNS peut faire autorité sur un domaine, c’est-à-dire que c’est lui qui a la réponse aux requêtes d’un domaine donné. Ainsi, le serveur d’autorité du domaine **com** peut légitimement renvoyer une réponse lorsqu’on lui parle de **zestedesavoir.com**.

Ce comportement est reproduit pour la plupart des domaines, jusqu’à arriver au serveur final qui nous délivrera la réponse attendue. Si vous demandez le domaine **fr.wikipedia.org** à un serveur DNS, en supposant qu’il ne le connaisse pas déjà, il va questionner un *root server* qui va le renvoyer vers le serveur d’autorité du domaine **org**. Ce dernier va renvoyer à son tour vers le serveur d’autorité de **wikipedia.org**. C’est finalement lui qui va fournir une ou plusieurs adresses IP associées à **fr.wikipedia.org**. On parle de structure en arbre inversé, avec la racine en haut. On peut observer ce fonctionnement en analysant une requête non récursive.



Lorsqu'on fait une requête DNS, par défaut, le serveur qu'on contacte transmet directement à son référent ou à un serveur d'autorité, et ainsi de suite. Cela permet d'instaurer un système de cache comme vu précédemment. C'est un fonctionnement **récuratif**. Il est possible d'effectuer des requêtes en mode **itératif (non récuratif)**, ce qui fait que le demandeur réémet une requête à chaque étape, au lieu de laisser les serveurs se contacter entre eux.

La commande suivante interroge le serveur DNS public 1.1.1.1 et effectue la requête de manière itérative.

```

1 # dig @1.1.1.1 fr.wikipedia.org +trace
2
3 ; <<>> DiG 9.10.3-P4-Debian <<>> @1.1.1.1 fr.wikipedia.org +trace
4 ; (1 server found)
5 ;; global options: +cmd
6 .                303      IN      NS      g.root-servers.net.
7 .                303      IN      NS      h.root-servers.net.
8 .                303      IN      NS      i.root-servers.net.
9 .                303      IN      NS      j.root-servers.net.
10 .               303      IN      NS      k.root-servers.net.
11 .               303      IN      NS      l.root-servers.net.
12 .               303      IN      NS      m.root-servers.net.
13 .               303      IN      NS      a.root-servers.net.
14 .               303      IN      NS      b.root-servers.net.
15 .               303      IN      NS      c.root-servers.net.
16 .               303      IN      NS      d.root-servers.net.
17 .               303      IN      NS      e.root-servers.net.
18 .               303      IN      NS      f.root-servers.net.
19 .               303      IN      RRSIG   NS 8 0 518400
                20190901050000 20190819040000 59944 . [...]
20 ;; Received 717 bytes from 1.1.1.1#53(1.1.1.1) in 5 ms
21
22 org.              172800  IN      NS
23   a0.org.afiliast.info.
24 org.              172800  IN      NS
25   a2.org.afiliast.info.
26 org.              172800  IN      NS
27   b0.org.afiliast.org.
28 org.              172800  IN      NS
29   b2.org.afiliast.org.
30 org.              172800  IN      NS
31   c0.org.afiliast.info.
32 org.              172800  IN      NS
33   d0.org.afiliast.org.
34 org.              86400   IN      DS      9795 7 1
35   364DFAB3DAF254CAB477B5675B10766DDAA24982

```

## VI. Reprenons du service

```
29 org. 86400 IN DS 9795 7 2
   3922B31B6F3A4EA92B19EB7B52120F031FD8E05FF0B03BAFCF9F891B
   FE7FF8E5
30 org. 86400 IN RRSIG DS 8 1 86400
   20190901170000 20190819160000 59944 . [...]
31 ;; Received 818 bytes from 198.97.190.53#53(h.root-servers.net) in
   80 ms
32
33 wikipedia.org. 86400 IN NS ns2.wikimedia.org.
34 wikipedia.org. 86400 IN NS ns0.wikimedia.org.
35 wikipedia.org. 86400 IN NS ns1.wikimedia.org.
36 h9p7u7tr2u91d0v0ljs9l1gidnp90u3h.org. 86400 IN NSEC3 1 1 1
   D399EAAB H9PARR669T6U801GSG9E1LMITK4DEM0T NS SOA RRSIG DNSKEY
   NSEC3PARAM
37 h9p7u7tr2u91d0v0ljs9l1gidnp90u3h.org. 86400 IN RRSIG NSEC3 7 2
   86400 20190909185601 20190819175601 44078 org.
   H6i2CguBU78xSKcGUIqL86LmOZP70F0gWJFicz6VCqp1phCqnSocrEP
   kp2vL9cgYuCZUB7uifT2JmmOmxdm336gBfFQxnSLU8wD5XDFthwZK/4
   gjAQL/H2qCKGd+uwlQXUqDA0emWZoMJq60eEftTU0mlucqQtHigRy1cs 994=
38 hhdkj29uji44onob8fm7tnl1n6nes0jb.org. 86400 IN NSEC3 1 1 1
   D399EAAB HHE40886E1IPI0A8N1TCCMQEN3JPUHJO NS DS RRSIG
39 hhdkj29uji44onob8fm7tnl1n6nes0jb.org. 86400 IN RRSIG NSEC3 7 2
   86400 20190907152407 20190817142407 44078 org.
   PcdtrdhckwwQweIf1aVvpr69kgQnw9dwJC/orqxTp3Ed4FwGpb9pY3b0
   jaXBW34bj0xNk9pdE2ysbHK1wZUa5eAipTcKr2jcAW8SIkplsN2/lF80
   b2+zCdgaMBACQhYXKfTw4AJICWJaYpGHJ5R6c8Qv1ipaCYXbvcDTwx05 v1I=
40 ;; Received 650 bytes from 199.19.53.1#53(c0.org.afiliat-nst.info)
   in 6 ms
41
42 fr.wikipedia.org. 86400 IN CNAME dyna.wikimedia.org.
43 ;; Received 74 bytes from 91.198.174.239#53(ns2.wikimedia.org) in
   22 ms
```

Quelques lignes ont été tronquées pour améliorer la lisibilité. On suit bien la logique décrite : d'abord on récupère auprès de 1.1.1.1 les adresses des *root servers*. Pour cela, on lui demande l'adresse de « . ». Oui, point. « . » représente le domaine d'origine, qui fait autorité pour tous les domaines de premier niveau.



Dans le système de noms de domaine, il existe une subtilité à laquelle on n'est confronté que lorsqu'on configure un serveur DNS. Quand un nom de domaine est écrit sans point à la fin, il est **relatif** au domaine dans lequel on se trouve. Ainsi, si vous configurez le domaine **agrume.com** pour qu'il redirige vers **zestedesavoir.com**, vous allez vous retrouver vers **zestedesavoir.com.agrume.com** ! Pour préciser que le domaine cible est **absolu**, et non relatif, il faut rajouter un point à la fin, pour signifier qu'on veut pointer sur **zestedesavoir.com**. (« relatif au domaine d'origine », donc absolu). Sur Internet, on n'est pas dans un domaine en particulier, donc tous les domaines qu'on demande sont absolus. On peut, si on le souhaite, rajouter un point à tous nos noms de domaines : ça marchera ! Essayez donc de faire une requête DNS sur **zestedesavoir.com.** avec le point



à la fin ! 🍌 Cette notation s'appelle **FQDN**, pour *Fully Qualified Domain Name*.

Ensuite, on demande à l'un des serveurs racines qui a autorité sur le domaine **org**. Dans notre cas, c'est le serveur racine H qui a été interrogé. Il existe 13 adresses pour les *root servers* : ce sont des sous-domaines de root-servers.net qui vont de A à M. [La page Wikipédia consacrée au sujet](#) [explique plus en détail comment ils fonctionnent](#).

Un des serveurs d'autorité du TLD **org** (dans notre cas, c0.org.afiliat-nst.info à l'adresse 199.19.53.1) est interrogé pour connaître qui contrôle le domaine **wikipedia.org**. Enfin, ns2.wikimedia.org (91.198.174.239) nous dit à quoi correspond **fr.wikipedia.org**.



dig n'a pas affiché les adresses IP récupérées à chaque étape. Pour interroger le serveur racine H, une autre requête a dû être effectuée pour connaître l'IP de h.root-servers.net, et ainsi de suite. Dans le résultat de la commande, à la fin, on nous dit que **fr.wikipedia.org** est équivalent à **dyna.wikimedia.org**. Nous allons apprendre à déchiffrer, entre autres, ces NS et CNAME dans la section suivante.

On peut représenter les échanges précédents par ce schéma.

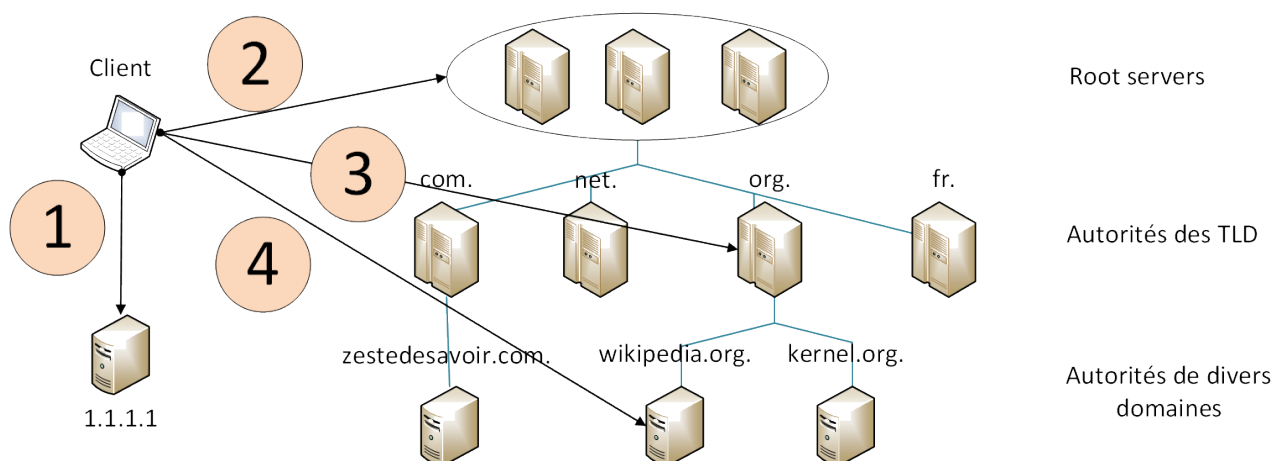


FIGURE VI.1.3. – Cheminement d'une requête DNS en mode itératif



Ce schéma ne représente que des serveurs DNS et un client, sans considération pour le réseau. Dans la réalité, ils ne sont pas directement connectés.

Nous avons vu dans cette section que des échanges DNS, ce n'est finalement pas que des associations de noms de domaine et d'adresses IP. Voyons maintenant les différents types de requêtes qui existent.

### VI.1.3. Drôle de type

Il existe officiellement plusieurs dizaines de types d'enregistrements et donc de requêtes DNS. Voici une liste de ceux qu'on rencontre le plus souvent.

### VI.1.3.1. Association d'adresses IP

C'est quand même l'intérêt principal du service : associer un nom de domaine avec une adresse IP. Les enregistrements de type **A** sont ceux qui renvoient des adresses IPv4. Autrement dit, quand on souhaite obtenir comme information une adresse IPv4, on envoie une requête de type A. Pour les adresses IPv6, c'est le type **AAAA**.

### VI.1.3.2. Services particuliers

Il y a un type d'enregistrement qui ne sert que pour les e-mails : il s'agit de **MX** (*Mail eXchange*). Les serveurs SMTP [☞](#) s'en servent pour savoir vers où transférer les messages. Quand un e-mail à destination de [truc@example.org](mailto:truc@example.org) [☞](#) doit être acheminé, le serveur SMTP envoie une requête DNS de type MX pour connaître l'adresse du serveur mail associé au domaine example.org. On trouve aussi des enregistrements **SRV** (**service**) qui permettent de renvoyer une adresse de serveur selon le service désiré. Ainsi, si on veut connaître l'adresse du serveur SIP (téléphonie sur IP) du domaine example.org, on peut demander le domaine `_sip.tcp.example.org`. Si l'enregistrement existe, il renverra l'adresse et le numéro de port du serveur associé. Toutefois, il ne faut pas trop compter dessus : ce type d'enregistrement est assez peu utilisé en pratique.

### VI.1.3.3. Allez voir ailleurs

L'enregistrement de type **CNAME** correspond à un alias. On envoie rarement une requête DNS pour obtenir effectivement un CNAME, mais on peut en recevoir quand on envoie une requête de type A ou AAAA. Une réponse CNAME signifie alors que le nom de domaine demandé renvoie vers un autre.

Il ne faut pas confondre avec le type **NS** (*Name Server*). Celui-ci indique que l'autorité du domaine demandé est transférée à un autre serveur. On peut aussi recevoir ce genre de réponse avec une requête de type A ou AAAA. Ce sont généralement les serveurs DNS qui reçoivent ce type de réponse, quand ils recherchent une adresse en remontant les domaines (org., example.org, ...). On a pu voir cela lors de l'exemple de la section précédente.

### VI.1.3.4. Divers

L'enregistrement **SOA** (*Start Of Authority*) est particulier. Il doit théoriquement être systématiquement défini pour une zone donnée. Une zone correspond à un sous-domaine, comme test.example.org ou hello.example.org. Il permet de délivrer des informations sur le maître de la zone, comme le serveur primaire ou une adresse e-mail de contact. Cela est utile quand plusieurs serveurs collaborent sur un domaine donné, par exemple pour se répartir la charge.

Terminons ce petit tour d'horizon des requêtes DNS par le type **TXT**. Pas de piège, ça veut bien dire **texte**. Si la [RFC 1035](#) [☞](#), qui définit ce standard, ne dit rien à son sujet, elle est en pratique utilisée à des fins d'authentification. Les enregistrements de type TXT permettent de renvoyer une information textuelle. Cela permet de prouver à des services que vous êtes bien le propriétaire d'un nom de domaine, sans en altérer le fonctionnement normal. Ils peuvent aussi être utilisés conjointement avec d'autres types de requêtes, comme **RP** (*Responsible Person*, assez rare), qui servent à demander qui est la personne responsable du domaine. Ne confondez pas avec SOA : on parle ici d'une personne, pas d'un serveur.

### VI.1.3.5. Récapitulatif

En résumé, les principaux types d'enregistrements DNS sont récapitulés dans le tableau suivant.

Type	Utilité	Fréquence d'utilisation
A	adresse IPv4	très courant
AAAA	adresse IPv6	très courant
MX	serveur SMTP	très courant
SRV	serveur pour un service donné	courant
CNAME	alias de nom	très courant
NS	serveur d'autorité	très courant
SOA	maitre d'une zone	très courant
TXT	texte	courant
RP	personne responsable du domaine	rare

### VI.1.4. Évolutions

DNS a évolué depuis sa création en 1983, mais fondamentalement, il s'agit toujours d'échanges en clair sur UDP. Avec l'explosion de l'utilisation d'Internet et des réseaux IP en général, des problèmes de sécurité ont commencé à apparaître. Seulement, on ne peut pas changer d'un claquement de doigts un système sur lequel le monde entier repose. Des améliorations sont apparues et sont progressivement adoptées depuis plusieurs années.

#### VI.1.4.1. DNSSEC

Dans les années 1990, pour apporter un peu de sécurité, une amélioration au protocole DNS a été proposée sous le nom *Domain Name System Security Extensions*, abrégé **DNSSEC**. Ce système a été ensuite modernisé en 2005. Il consiste à signer les enregistrements DNS émis pour éviter qu'ils ne soient altérés. Techniquement, pour reprendre notre cas de serveur DNS d'opérateur, rien ne nous garantit que les enregistrements qu'il retourne soient les vrais et qu'il ne les modifie pas comme bon lui semble. Dans les faits, de telles altérations sont opérées par des opérateurs de télécoms à la demande de gouvernements ou suite à des décisions de justice dans des pays qui pratiquent la censure du net, comme la Chine, l'Irlande ou la France (voir [cet article en anglais](#) [↗](#) pour plus d'informations). On peut aussi s'inquiéter, sur un réseau local, d'une attaque [Man In The Middle](#) [↗](#).



C'est en partie pour disposer d'une indépendance par rapport aux serveurs DNS d'opérateurs télécoms qu'il existe des services indépendants comme [OpenDNS](#) [↗](#), [Google Public DNS](#) [↗](#) ou encore [le DNS de Cloudflare](#) [↗](#).

## VI. Reprenons du service

Avec DNSSEC, si des enregistrements DNS signés sont altérés, ça se sait tout de suite : la signature n'est plus bonne. Pour plus d'informations sur les mécanismes de signature numérique, nous vous invitons à consulter [ce document de l'université Paris-Est Marne-la-Vallée](#) . Ce système, bien que standardisé, n'est pas utilisé et déployé partout. L'intégralité des équipements sur Internet utilise DNS, alors on ne peut pas forcer le changement d'un des plus fondamentaux services du réseau. Cela se fait progressivement. Les systèmes d'exploitation modernes le supportent nativement.



La signature ne chiffre pas le contenu des échanges : requêtes et réponses sont toujours transmises en clair sur le réseau.

### VI.1.4.2. DOT

Une autre façon de se prémunir d'une altération en cours de route est l'utilisation de *DNS Over TLS* (**DOT**). Le principe est simple : les échanges DNS sont encapsulés dans un tunnel TLS entre le client et le serveur. Cela implique nécessairement davantage d'échanges pour initier la connexion sécurisée, d'autant qu'ils sont transportés par TCP. Aussi, vu qu'il s'agit d'un protocole différent du DNS *vanilla* (c'est-à-dire "d'origine", "pur"), un port différent doit lui être attribué. En l'occurrence, c'est le port 853 qui remplit cette fonction. Cela ajoute une difficulté dans certains environnements sécurisés ou filtrés : il est courant, dans des réseaux d'entreprise, d'empêcher tout trafic non prévu, ce qui fait que si l'utilisation du DOT n'est pas explicitement admise, ce nouveau port risque d'être bloqué par défaut. Il en va également ainsi pour certains accès publics, comme des réseaux Wi-Fi de restaurants, qui peuvent souhaiter empêcher l'utilisation de services inconnus.

Certains serveurs DNS publics supportent DOT, mais son utilisation n'est pas encore très courante. Cette technique a été standardisée en 2016, puis améliorée en 2018. Android le supporte nativement depuis sa version Pie. Windows 10 ne permet pas son utilisation sans un logiciel tiers comme Chocolatey. Pour plus d'informations techniques, vous pouvez consulter les documents de référence : la [RFC 7858](#) et la [RFC 8310](#) .

### VI.1.4.3. DOH

Il est aussi possible d'encapsuler des échanges DNS dans HTTPS avec le *DNS Over HTTPS* (**DOH**). Cette proposition est publiée notamment par Mozilla en octobre 2018 dans la [RFC 8484](#) . Elle permet de contourner d'éventuels filtrages en ayant recours à un protocole sécurisé extrêmement commun et rarement filtré (HTTPS), mais cela nécessite que le client supporte ce système. Cela ajoute aussi de l'encapsulation et donc réduit potentiellement les performances d'un point de vue réseau. En 2024, DOH est supporté nativement par les systèmes d'exploitation Windows 11, iOS, macOS et Android mais son utilisation nécessite un paramétrage spécifique.

## Conclusion

DNS est un système complexe, d'importance vitale et dont les évolutions sont difficiles à mettre en place. Son omniprésence empêche une mise à jour globale, car tous les systèmes doivent continuer à fonctionner, ce qui impose aux serveurs une compatibilité constante et cause une inertie importante au déploiement à grande échelle.

Pour approfondir le sujet, nous vous recommandons les ressources suivantes :

## VI. Reprenons du service

- [How DNS works](#) , une bande dessinée en anglais qui illustre le fonctionnement de DNS ;
- [Protocole DNS](#) , article de FrameIP en français qui se focalise sur l'aspect protocole ;
- [DNS Zones Explained](#) , un article en anglais qui explique ce qu'est une zone DNS et qui revient sur les points importants de ce chapitre.

Enfin, les spécificités de ce protocole, notamment son mode récursif et sa capacité à transporter autre chose que des adresses IP, permettent d'en détourner l'usage initial. Ainsi, il est techniquement possible d'encapsuler le protocole IP lui-même dans DNS ! Cette prouesse est appelée [IP over DNS](#) . Les performances sont extrêmement limitées et les cas d'usage restreints, mais l'exploit mérite d'être mentionné. Plus modestement, [le système Chatavion](#) permet, grâce à une application Android, l'échange de messages courts type SMS au travers du protocole DNS. Cela permet une communication minimale sur des réseaux limités, par exemple en 2G, et même sur des accès Wi-Fi avec portail captif.

## VI.2. DHCP, un service rempli d'adresses

### Introduction

Avec un titre pareil, on se demande si on est sur un court de tennis ou un cours de réseau.

🍊 C'est pourtant bien pour pouvoir communiquer que DHCP est indispensable dans la vie quotidienne. Si vous n'avez pas paramétré votre adresse IP pour lire ce chapitre, c'est très certainement grâce à lui !

Ce protocole sert à distribuer automatiquement des configurations IP dès que vous connectez votre appareil à un réseau. Nous allons d'abord voir comment il fonctionne globalement, puis nous nous intéresserons aux possibilités de configuration offertes par les routeurs et les serveurs.

### VI.2.1. Côté client

DHCP est un protocole de niveau 7 qui permet d'obtenir une adresse IP automatiquement, sans avoir à configurer sa carte réseau. C'est bien pratique quand on ne connaît pas le réseau, et surtout, on ne risque pas de s'attribuer par erreur une adresse déjà utilisée.

C'est bien pratique aussi pour l'administrateur réseau qui n'a pas envie de configurer les 500 machines de son entreprise. En plus, en cas de changement de plan d'adressage, il aura encore moins envie de devoir refaire tout le travail. 🍊

Vous l'avez sûrement remarqué, sous la plupart des systèmes d'exploitation (Windows, Android, ...), la configuration réseau se fait automatiquement. Elle est généralement fournie par DHCP. Voyons justement comment cela fonctionne pour nous, clients. Que se passe-t-il quand on branche un câble dans la prise, ou quand on se connecte sur un réseau Wi-Fi ?

Nous avons dit que DHCP est de niveau 7. Il fonctionne au dessus d'UDP et IP.

...

Il n'y a pas quelque chose qui vous choque ? 🍊

?

DHCP sert à obtenir une adresse IP, donc on l'utilise quand on n'en a pas, donc... comment ça peut fonctionner sur IP !?

Hé hé hé... Il y a deux cas différents, selon qu'on utilise IPv4 ou IPv6.

#### VI.2.1.1. En IPv4

En réalité, c'est tout bête : quand une machine demande une adresse IP en DHCP, elle prend l'adresse 0.0.0.0.

L'obtention d'une adresse IP se fait en plusieurs phases :

1. Le client envoie en broadcast une recherche de serveur DHCP (*DHCP discover*, découverte). Niveau transport, le port source est toujours 68 et le port destination 67. Cela

## VI. Reprenons du service

implique nécessairement que la machine qui répond soit sur le même réseau physique : les broadcasts ne sont pas propagés par les routeurs.

2. Tous les serveurs DHCP ayant reçu la requête et ayant une offre à proposer y répondent en envoyant leur adresse IP ainsi que la configuration (adresse, masque) qu'ils proposent au client (*DHCP offer*, offre). Pour cela, ils utilisent l'identifiant de niveau 2 de l'émetteur, comme l'adresse MAC sur un réseau Ethernet, afin que la réponse arrive au bon demandeur.
3. Le client ne retient que la première réponse qui lui parvient, s'approprie la configuration retenue et envoie en broadcast l'adresse du serveur choisi pour qu'il réserve l'adresse et pour que les autres sachent que leur proposition a été rejetée (*DHCP request*, requête).
4. Le serveur confirme et informe le client de la durée de validité de cette adresse (le bail). En complément, il peut aussi lui indiquer une passerelle par défaut, des serveurs DNS, ... Cette dernière phase est appelée *acknowledgment* (acceptation).



Discover. Offer. Request. Acknowledgment. Découverte. Offre. Requête. Acceptation. Si on garde les initiales de chaque mot, ça fait DORA. Voilà un bon moyen mnémotechnique pour se souvenir des 4 phases ! 🧙🏻

## VI. Reprenons du service

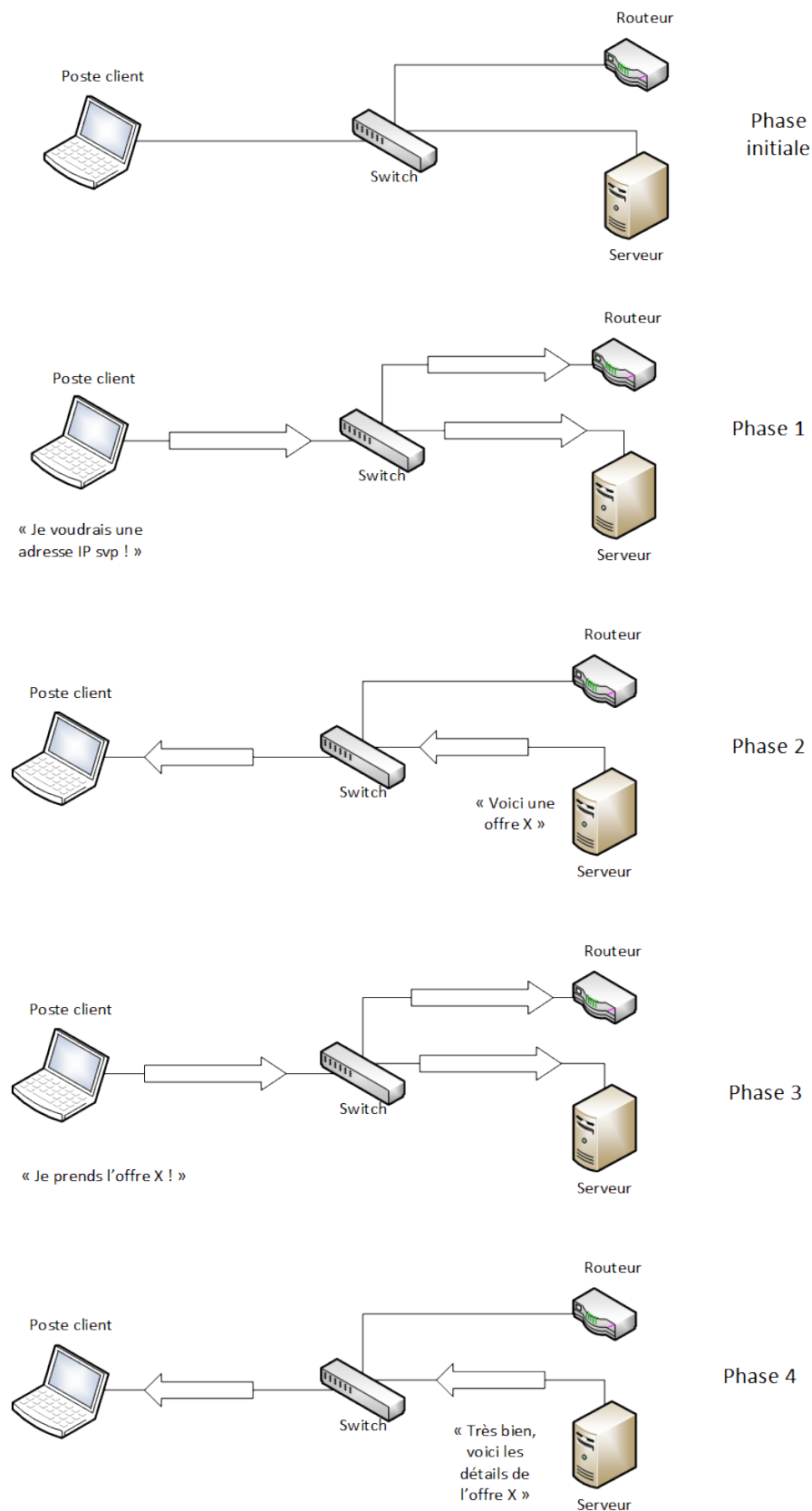


FIGURE VI.2.1. – Les 4 phases d'attribution d'une adresse IP en DHCP

Ces détails peuvent être visualisés à l'aide d'un [analyseur de trames](#) .

## VI. Reprenons du service

No.	Time	Source	Destination	Protocol	Length	Info
13	16.209490	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xac7ee63
23	16.863676	10.63.255.252	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0xac7ee63
24	16.865092	0.0.0.0	255.255.255.255	DHCP	370	DHCP Request - Transaction ID 0xac7ee63
27	17.096591	10.63.255.252	255.255.255.255	DHCP	342	DHCP ACK - Transaction ID 0xac7ee63

FIGURE VI.2.2. – Capture de trames des 4 phases d’une transaction DHCP

On peut se pencher en détail sur les différentes couches pour chacune de ces phases.

```
> Frame 13: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
> Ethernet II, Src: Intel[redacted]:8a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 68, Dst Port: 67
> Dynamic Host Configuration Protocol (Discover)
```

FIGURE VI.2.3. – Phase 1 d’une transaction DHCP

On visualise ici l’utilisation de l’adresse 0.0.0.0, l’envoi en broadcast niveau IP et Ethernet, ainsi que les ports UDP utilisés.

```
> Frame 23: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
> Ethernet II, Src: FreeboxS_00:01:00 (00:07:cb:00:01:00), Dst: Intel[redacted]:8a)
> Internet Protocol Version 4, Src: 10.63.255.252, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 67, Dst Port: 68
> Dynamic Host Configuration Protocol (Offer)
```

FIGURE VI.2.4. – Phase 2 d’une transaction DHCP

Notons que lors de la 2ème phase, la réponse est adressée en unicast au demandeur au niveau 2, mais au niveau 3, c’est l’adresse de broadcast qui est utilisée.

```
> Frame 24: 370 bytes on wire (2960 bits), 370 bytes captured (2960 bits) on interface 0
> Ethernet II, Src: Intel[redacted]:8a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 68, Dst Port: 67
> Dynamic Host Configuration Protocol (Request)
```

FIGURE VI.2.5. – Phase 3 d’une transaction DHCP

Là encore, cette trame est émise en broadcast.

```
> Frame 27: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
> Ethernet II, Src: FreeboxS_00:01:00 (00:07:cb:00:01:00), Dst: Intel[redacted]:8a)
> Internet Protocol Version 4, Src: 10.63.255.252, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 67, Dst Port: 68
> Dynamic Host Configuration Protocol (ACK)
```

FIGURE VI.2.6. – Phase 4 d’une transaction DHCP

Et voilà, l’attribution de l’adresse est confirmée ! 🍊

### VI.2.1.2. En IPv6

Avec IPv6, toute carte réseau dispose de base d'une adresse de lien local, générée par le système d'exploitation. DHCP peut tout de même servir à obtenir une adresse globale. Le principe est sensiblement le même qu'avec IPv4 :

1. Le client sollicite un serveur DHCP (solicit). Niveau réseau, l'adresse IP source est celle de lien local. L'adresse de destination est FF02::1:2. Les broadcasts n'existant pas en IPv6, c'est cette adresse de multicast qui regroupe tous les serveurs DHCP. Niveau transport, le port source est 546, le port destination est 547.
2. Tous les serveurs atteints répondent par une annonce, s'ils ont une offre à proposer (advertise).
3. Le client confirme la demande qui lui convient (la première reçue, généralement) au moyen d'un paquet émis sur le même multicast qu'à l'étape 1 (request).
4. Le serveur confirme l'attribution et fournit éventuellement des informations complémentaires (reply).



Lorsqu'on parle de DHCP sur IPv6, on le précise souvent en parlant de **DHCPv6**. Ce n'est pas le seul à fournir des adresses : le protocole **NDP** propose aussi ce service.

Ça, c'est ce que voit un client. Mais de l'autre côté du réseau, les choses peuvent être plus complexes que cela.

### VI.2.2. Côté serveur

Dans les cas les plus simples, c'est un routeur qui fait office de serveur DHCP. Il n'en a toutefois pas l'obligation, il peut même ne pas proposer ce service du tout. Le rôle d'un routeur, ce n'est pas de fournir des adresses : quand on doit assurer le routage de données de nombreux flux, on a parfois autre chose à faire que de distribuer des IP. 🍊

#### VI.2.2.1. Fonctionnement des serveurs

Cette tâche peut être dévolue à des serveurs, dont c'est déjà davantage le rôle. Un serveur DHCP est configuré avec une liste d'adresses qu'il peut attribuer et leur masque associé. On appelle cela un **pool**. On le paramètre aussi pour renvoyer d'autres informations au client, comme une adresse de passerelle par défaut ou des serveurs DNS.

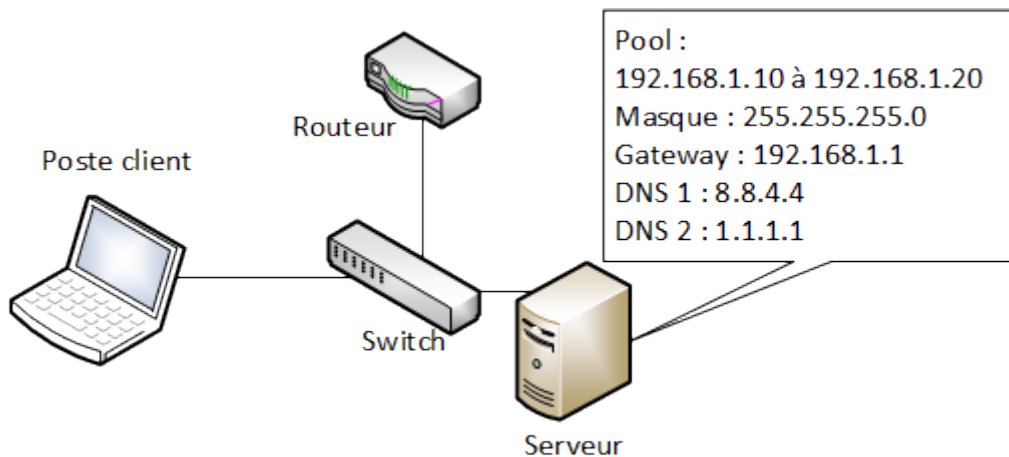


FIGURE VI.2.7. – Ce que "voit" un serveur DHCP

Quand un serveur DHCP reçoit une requête, il pioche une adresse dans ce pool (généralement, la première disponible dans l'ordre numérique) pour la fournir au client. Il est possible de configurer le serveur pour renvoyer une adresse IP spécifique en fonction de l'adresse physique (MAC) du demandeur. L'adresse IP fournie est enregistrée comme étant prise par le serveur et ne peut plus être distribuée. Cette attribution est limitée dans le temps par un **bail** qui peut être défini comme on le souhaite : 2 heures, 1 jour, une semaine... Pour pouvoir continuer à utiliser l'adresse fournie, le client doit formuler une demande de renouvellement de bail à la moitié du temps de validité. Ainsi, si le bail est de 2 heures, le client doit formuler une demande au bout d'une heure pour le prolonger. Sans renouvellement, le serveur demande au client à la fin du bail s'il souhaite conserver son adresse. En l'absence de réponse, elle est libérée et peut de nouveau être attribuée.

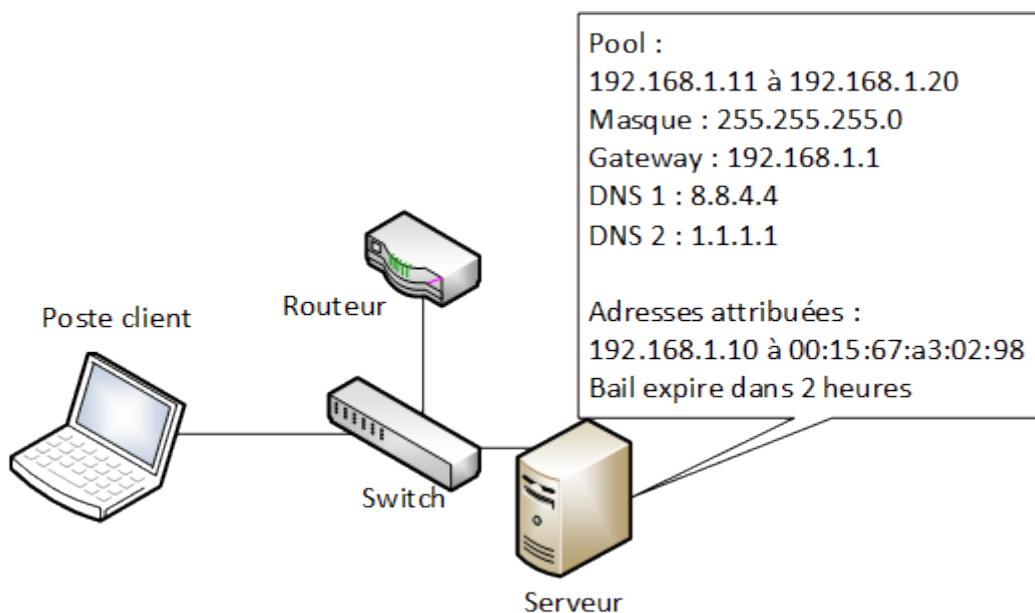


FIGURE VI.2.8. – Ce que "voit" un serveur DHCP après attribution d'une adresse



Si toutes les adresses sont attribuées et que le pool est vide, le serveur ne répond plus aux nouvelles demandes.

## VI. Reprenons du service

Comme DHCP fonctionne grâce à des broadcasts, du moins avec IPv4, cela implique qu'il faut un serveur par réseau de niveau 2 (physique ou VLAN). Dans les entreprises, il peut y en avoir beaucoup, et mettre en place autant de serveurs DHCP, c'est contraignant ! On passe alors par des **agents relais**.

### VI.2.2.2. Agent à votre service

Pour centraliser la gestion des adresses IP, on peut tout configurer sur un seul serveur. Ensuite, des serveurs ou des routeurs peuvent être paramétrés sur chaque LAN pour servir de relai. On les appelle des agents relais, ou simplement des relais.

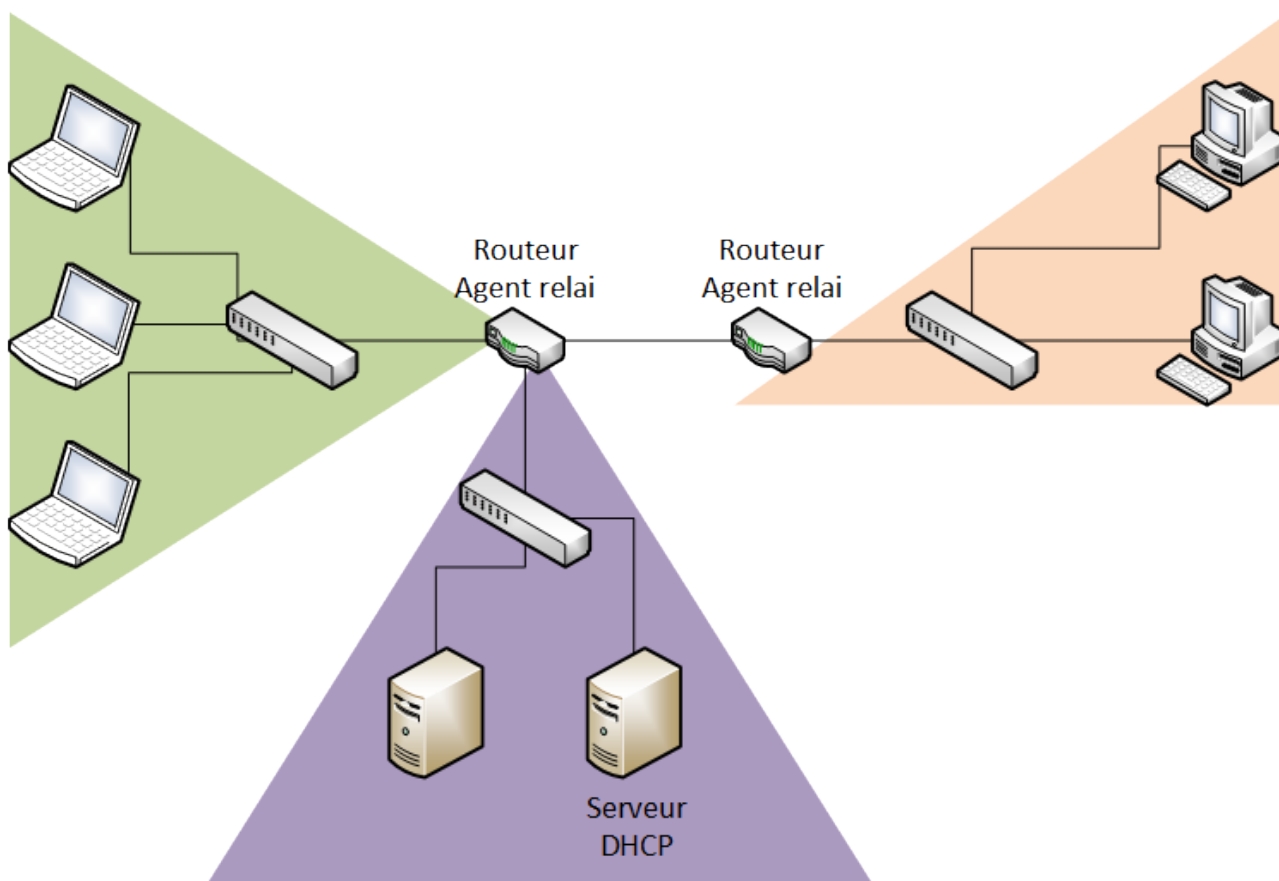


FIGURE VI.2.9. – Un broadcast émis dans une zone de couleur ne dépassera pas sa zone

Comme leur nom l'indique, ils relaient les flux DHCP jusqu'au serveur défini. C'est transparent pour le client : il ne voit que les échanges avec le relai comme si c'était le serveur final.

La gestion d'un grand nombre d'adresses peut même être compliquée pour un simple serveur DHCP sous Linux, par exemple. Il existe des solutions comme Efficient IP qui permettent de centraliser la gestion de quantités d'adresses et de simplifier le paramétrage dans des réseaux d'envergure.

## Conclusion

Dans la famille des services réseaux indispensables, DHCP mérite largement sa place sur le podium. Dans le prochain chapitre, nous nous pencherons sur un autre service fondamental dans les réseaux actuels.

## VI.3. NAT ou l'art de la dissimulation

### Introduction

Lors de la partie sur l'adressage, nous avons évoqué les adresses privées et le fait qu'elles ne peuvent pas être routées sur Internet. Pour pouvoir communiquer dans un tel cas, on peut faire de la **translation**. Ne fuyez pas tout de suite : ce n'est pas de la géométrie ! 🍊

#### VI.3.1. NAT : cette adresse que je ne saurais voir

La **translation d'adresse**, ou **NAT** pour *Network Address Translation*, est une technique qui consiste pour un routeur (ou un pare-feu) à remplacer une adresse IP source par une autre. L'hôte de destination ne voit que l'adresse de substitution.

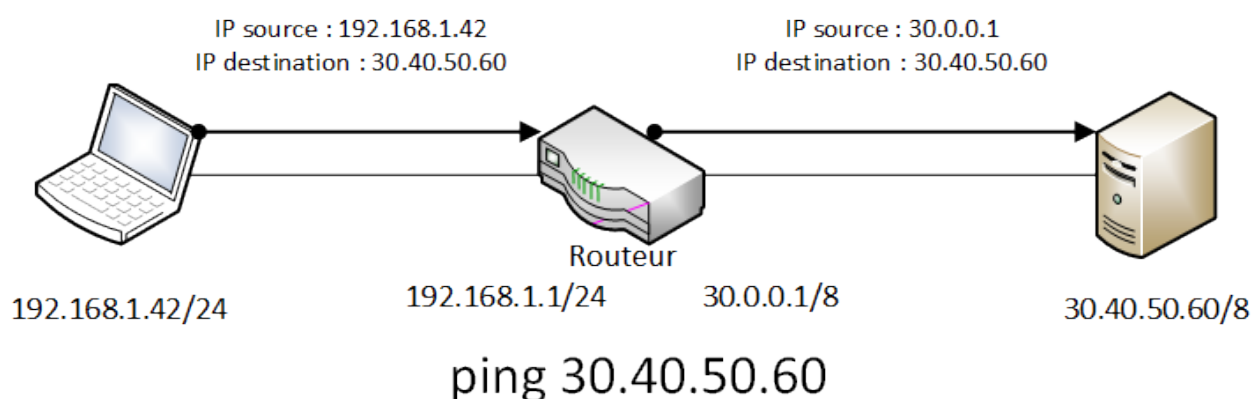


FIGURE VI.3.1. – Translation d'adresse

?

Comment il fait pour répondre ? 🍊

Eh bien, il répond à l'adresse source qu'il voit, le plus naturellement du monde. Elle doit normalement être attribuée au routeur qui a réalisé la translation d'adresse et qui va donc recevoir la réponse, et la renvoyer à l'hôte qui a émis la requête à l'origine.

?

Comment le routeur sait à qui renvoyer la réponse ?

Dans le cas le plus simple, on associe une adresse source réelle, **interne**, à une adresse de substitution, **externe**. Selon l'IP à qui est adressé le paquet, le routeur sait donc vers quel hôte le renvoyer. On appelle cela du **NAT statique**. Il est aussi possible de définir un pool d'adresses,

## VI. Reprenons du service

comme avec DHCP, et de piocher dedans quand il y a besoin. C'est du **NAT dynamique**. Utile quand on a moins d'adresses externes que d'adresses internes.

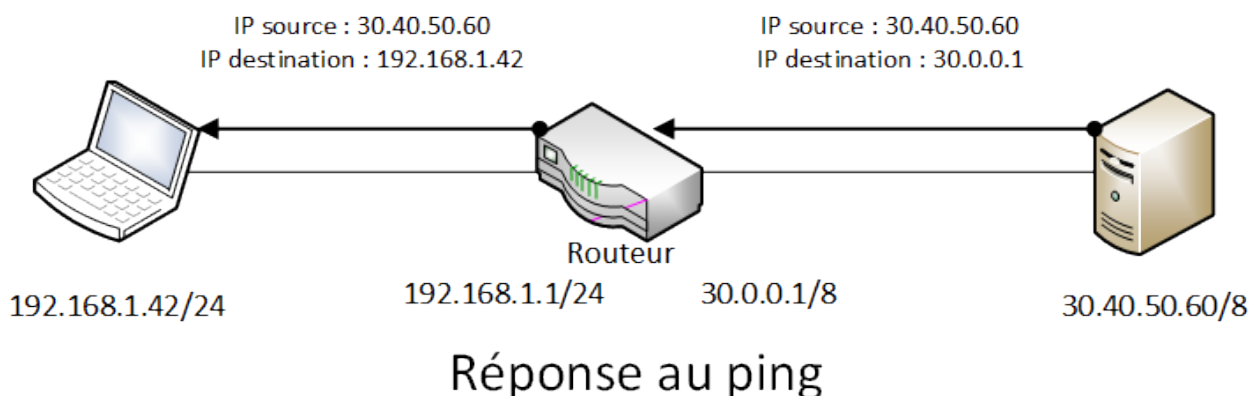


FIGURE VI.3.2. – Réponse à un paquet ayant subi une translation d'adresse

?

Ok... Et ça sert à quoi ?

Le principal intérêt est la sécurité. La translation d'adresse permet de dissimuler un hôte réel derrière un équipement qui se fait passer pour lui. En cas de tentative d'attaque, pas moyen de parvenir jusqu'à lui !

Pour être précis, il est techniquement possible de l'atteindre avec son adresse IP réelle, encore faut-il la connaître. Mais si elle n'est pas routée, aucun chemin ne permet de l'atteindre. 🧙‍♂️ C'est une des raisons pour lesquelles la plupart des fournisseurs d'accès à Internet, dans leurs routeurs, attribuent des adresses privées sur le réseau local et font du NAT. Les systèmes les plus vulnérables sont ainsi moins exposés sur Internet.

Une autre raison est que cela permet de faire des économies d'adresses publiques. Le stock d'adresses IPv4 disponibles sur Internet est limité, il n'y en a même pas une pour deux habitants sur Terre (environ 3,65 milliards d'adresses publiques pour 7,5 milliards d'humains en 2019). Alors avec la quantité d'objets connectés, si on veut continuer à utiliser IPv4, on rationalise encore davantage en faisant de la **translation de port**.

### VI.3.2. PAT: lost in translation

Quand on utilise un routeur fourni par notre opérateur télécom pour accéder à Internet, on peut y connecter de nombreux équipements en simultané, mais on ne dispose que d'une seule adresse IP publique (et donc externe). Pas moyen de se dire que l'adresse interne X va correspondre à l'adresse externe Y, puisque tout le monde a besoin de l'utiliser en même temps !

Alors, comment fait-on ? On va se baser à la fois sur les adresses de destination et sur les numéros de ports utilisés. Prenons un cas simple : deux équipements différents, un smartphone et un PC, se connectent sur deux sites web différents en même temps, disons Zeste de Savoir (92.243.7.44) et Wikipédia (91.198.174.192). Facile à NATer : le routeur peut retenir que la connexion vers Zeste de Savoir est associée au smartphone, et celle vers Wikipédia, au PC.

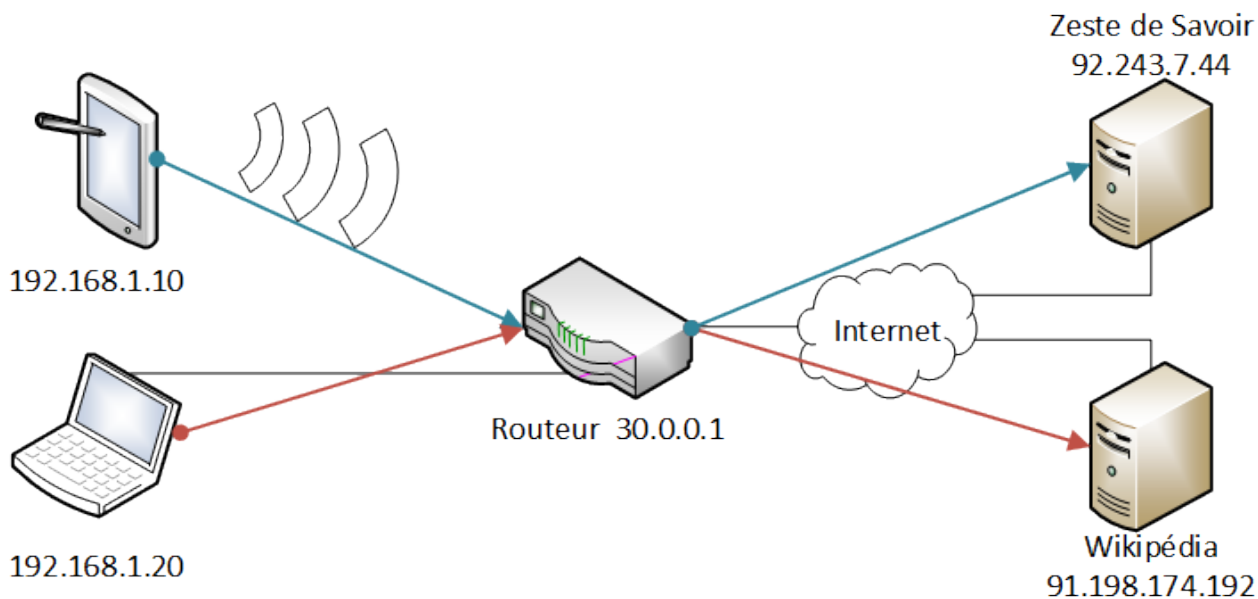


FIGURE VI.3.3. – Translations d'adresses avec plusieurs hôtes

Source réelle	Source après NAT	Destination (adresse : port)
192.168.1.10	30.0.0.1	92.243.7.44 : 443
192.168.1.20	30.0.0.1	91.198.174.192 : 443

Maintenant, que se passe-t-il si les deux appareils se connectent en même temps sur Wikipédia ? Là, le routeur va devoir en plus mémoriser les numéros de port source pour renvoyer les réponses aux bons hôtes. C'est ce qu'on appelle de la **translation de port**, abrégé **PAT** pour *Port Address Translation*.

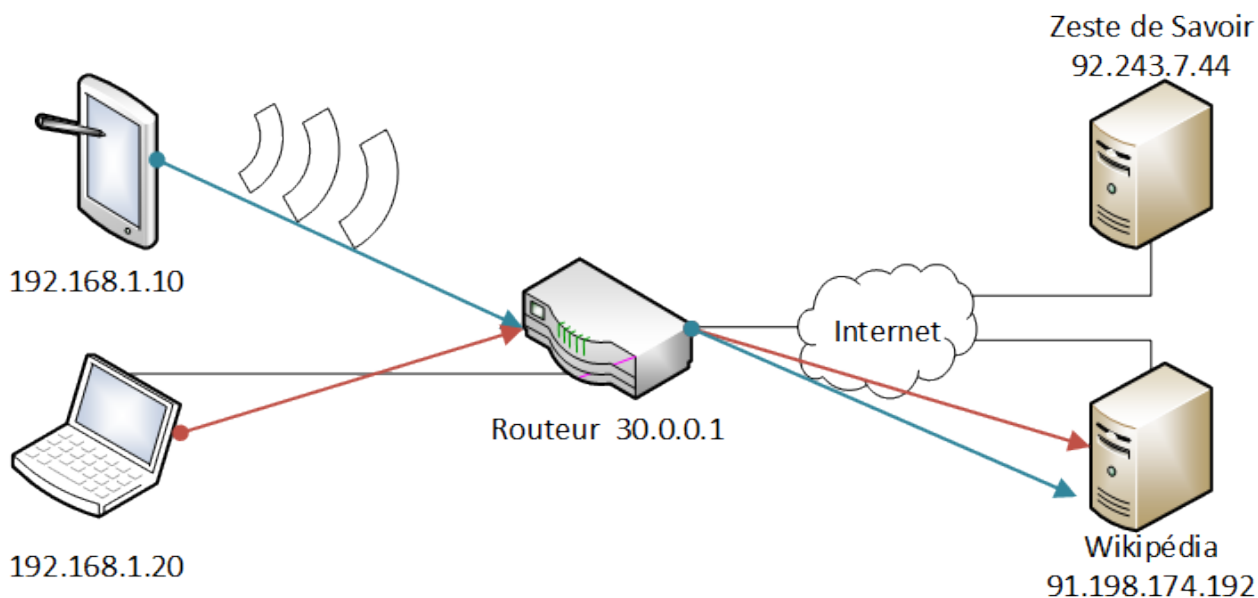


FIGURE VI.3.4. – Translation de ports

Source réelle	Source après NAT	Destination (adresse : port)
192.168.1.10 : 50000	30.0.0.1 : 50000	91.198.174.192 : 443
192.168.1.20 : 60000	30.0.0.1 : 60000	91.198.174.192 : 443

i

On trouve beaucoup de noms différents pour parler de PAT : NAPT (*Network Address and Port Translation*), *NAT overload* dans les routeurs Cisco, *IP masquerade* sous Linux avec iptables, ... voire tout simplement NAT, par abus de langage.

Le cas le plus complexe est quand plusieurs équipements du LAN se connectent en même temps au même serveur, au même service, et avec le même port source. Statistiquement, c'est très improbable, mais pas impossible ! Dans ce cas, le routeur modifie un des ports sources pour éviter d'avoir deux flux identiques qu'on ne pourrait plus différencier.

Source réelle	Source après NAT	Destination (adresse : port)
192.168.1.10 : 50000	30.0.0.1 : 50000	91.198.174.192 : 443
192.168.1.20 : 50000	30.0.0.1 : 56789	91.198.174.192 : 443

i

Pour NATer des flux ICMP, comme c'est un protocole de couche réseau, on ne peut bien évidemment pas utiliser de numéro de port. On prend à la place l'identifiant du paquet ICMP.

### VI.3.3. Le port forwarding

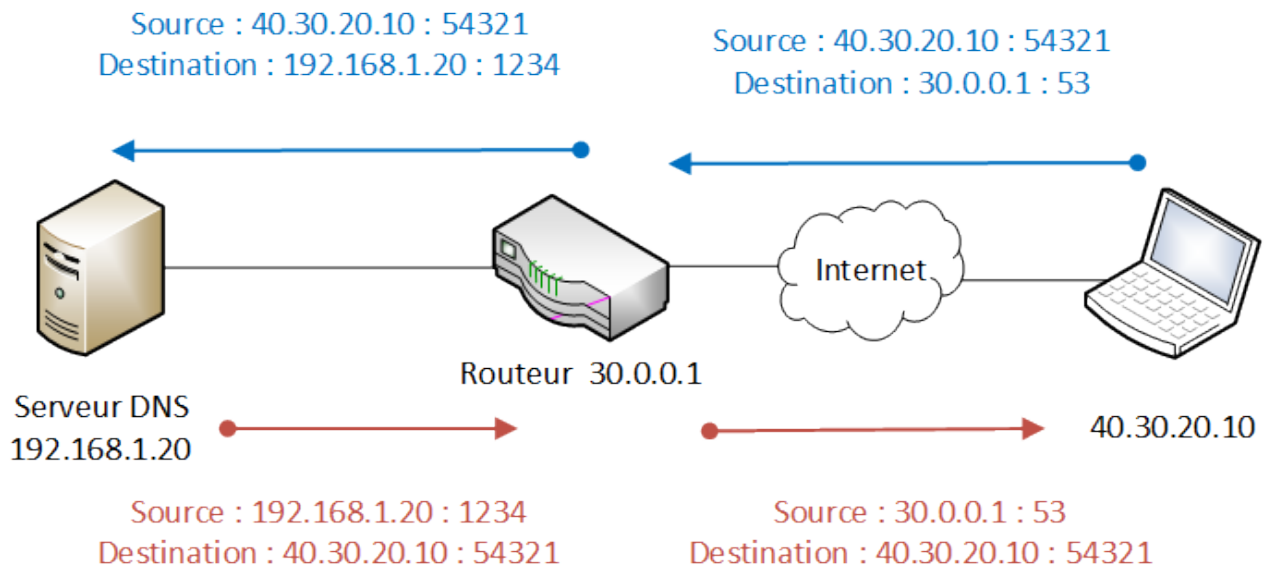
Le NAT permet donc de protéger un réseau, mais le revers de la médaille, c'est qu'on ne peut pas du tout envoyer de données depuis l'extérieur vers ce réseau ! On peut alors avoir recours au **port forwarding**.

Le port forwarding, c'est l'inverse du PAT. Quand un flux arrive à destination d'une adresse externe sur un port donné, le routeur vérifie dans une table de correspondance si ce port est ouvert. Si oui, il transmet le paquet à un hôte défini sur un port défini par sa configuration. Le schéma suivant représente un serveur DNS écoutant sur le port 1234, derrière un routeur qui fait du port forwarding. Il est interrogé par un hôte quelconque sur Internet. Le routeur est paramétré pour que les flux UDP à destination du port 53 soient redirigés vers ce serveur sur le port 1234.

i

Ce numéro de port est pris au hasard. C'est pour illustrer le fait qu'on peut utiliser le port qu'on veut sur le routeur, indépendamment du port réellement utilisé par le serveur. On aurait aussi très bien pu garder le port 53.

## Requête DNS



## Réponse DNS

FIGURE VI.3.5. – Port forwarding avec un serveur DNS

Cette technique est pratique quand on veut rendre accessible un serveur chez soi mais qu'on est derrière un routeur qui fait du NAT.

## Conclusion

La translation d'adresse est donc une technique utile pour la sécurité de réseaux locaux et l'économie d'adresses. Elle est principalement utilisée avec IPv4 mais rien n'empêche de faire du NAT avec IPv6. Elle ne doit pas non plus être considérée comme la panacée pour se protéger : l'utilisation d'un [pare-feu](#) bien configuré est plus efficace et moins contraignante.

## VI.4. Le proxy dans tous ses états

### Introduction

Si vous êtes en entreprise, à l'université ou à l'école, vous utilisez fort probablement un **proxy**. Vous avez peut-être déjà eu affaire à lui en essayant de naviguer sur un site web dont l'accès vous a été refusé. Nous allons dans un premier temps définir de quoi il s'agit concrètement. Ensuite, nous verrons son fonctionnement d'un point de vue réseau. Puis nous nous intéresserons à la place qu'il occupe dans les réseaux d'entreprise. Enfin, si vous êtes sages, vous aurez le droit à quelques pistes pour contourner cette interdiction injuste d'aller sur des sites de jeu au bureau.



### VI.4.1. Principe

Un **proxy**, appelé parfois **mandataire**, est un service réseau qui consiste à faire le relai entre un client et un serveur.

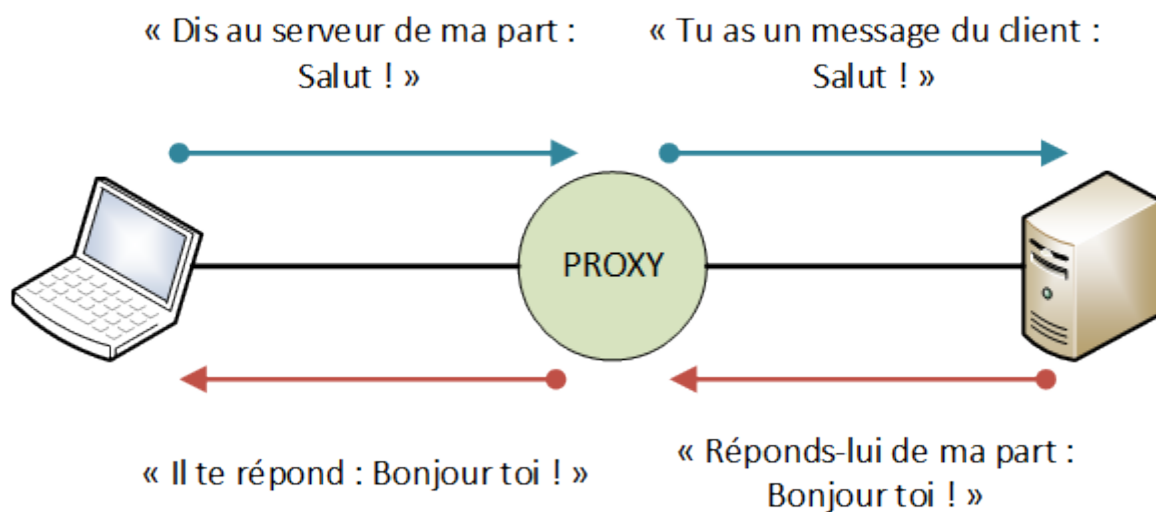


FIGURE VI.4.1. – Échange simple au travers d'un proxy

Sur cette image, l'information **utile** que le client fait parvenir au serveur est "Salut!". Mais le message qui parvient au serveur n'est pas "Salut!", c'est "Tu as un message du client : Salut!". Le proxy, en tant qu'intermédiaire, a la possibilité de modifier le contenu des échanges, pour ajouter des informations sur l'expéditeur notamment.



Ça sert à quoi ? Le client aurait aussi bien pu adresser son message au serveur sans intermédiaire.

Pas forcément ! Il peut y avoir plein de raisons pour lesquelles un proxy est utile. Par exemple, un serveur peut n'accepter de répondre qu'à des requêtes préalablement filtrées, pour éviter des sollicitations inutiles ou provenant d'inconnus. Le proxy sert alors de filtre en amont du serveur. Il peut aussi répondre à la place du serveur dans certains cas : quand ce dernier est harcelé de requêtes identiques à qui il doit répondre la même chose, le proxy peut répondre à sa place pour le décharger. C'est un système de **cache**, on parle alors de **proxy-cache**.

Dans la configuration dont nous parlons, le client contacte spécifiquement un proxy dans le but qu'il transmette un message au serveur. Autrement dit, le proxy est au service du client. Il existe un autre cas de figure, où le proxy est au service du serveur, et se fait passer pour lui. Là, c'est transparent pour le client, qui croit s'adresser en direct au serveur, alors que c'est un intermédiaire qui lui répond. Dans ce cas, on parle de **reverse proxy**.

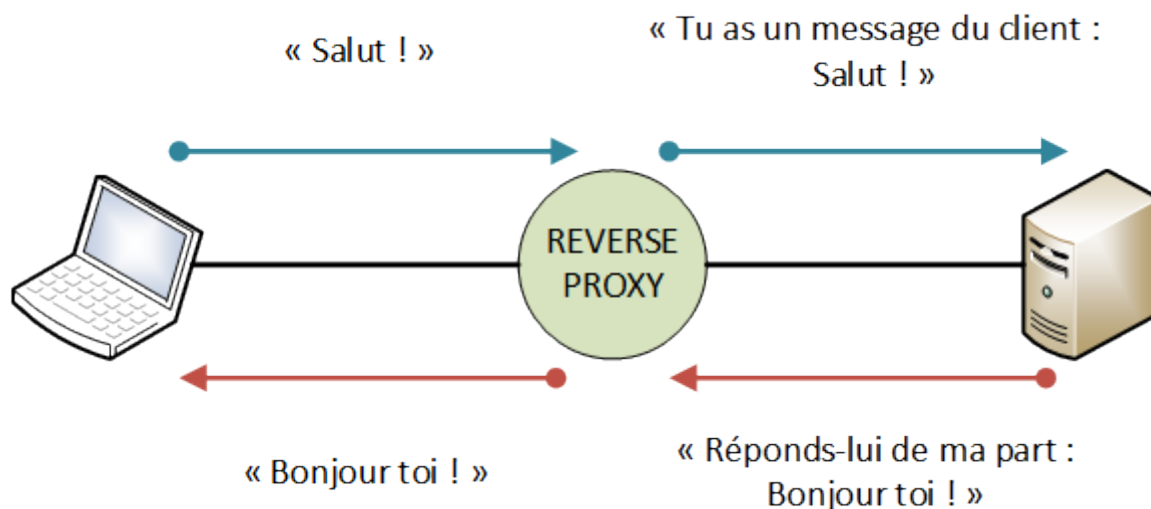


FIGURE VI.4.2. – Échange au travers d'un reverse proxy

En pratique, on peut utiliser un proxy pour plein de protocoles applicatifs différents : HTTP, IRC, SMTP, ... Le plus courant est le proxy web, qui relaie donc des requêtes et des réponses HTTP. Voyons comment cela se passe d'un point de vue réseau.

### VI.4.2. Aspect réseau

Concrètement, un proxy est une application, un logiciel. Parmi les plus connus, on peut citer Squid pour le web, ou Firewall de Watchguard. Il fonctionne au niveau 7 du modèle OSI, c'est-à-dire sur la couche application. Si vous avez bien suivi le début du cours, vous vous souvenez que pour atteindre le niveau le plus haut, il faut décapsuler toutes les couches inférieures. C'est extrêmement important de garder cela en tête, car quand un proxy web reçoit une requête HTTP et veut la transmettre à un serveur, la réception se fait d'un côté avec une connexion TCP vers le client, des paquets IP dont il est le destinataire... Et la transmission d'un autre côté, avec une autre connexion TCP vers le serveur et des paquets IP dont il est la source ! Il y a une rupture de flux au niveau du proxy.

## VI. Reprenons du service

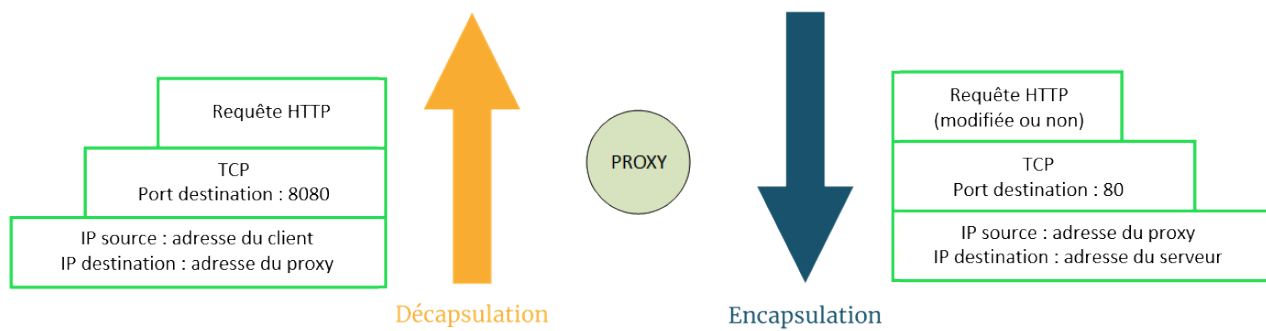


FIGURE VI.4.3. – Décapsulation de la requête reçue et réencapsulation

Observons cela avec une analyse réseau depuis un proxy.

No.	Time	Source	Destination	Protocol	Length	Info
18	1.553071	66.66.207.207	51.17.17.204	HTTP	462	GET http://forumactif.com/ HTTP/1.1
23	1.580040	51.17.17.204	188.165.2.137	HTTP	538	GET / HTTP/1.1
41	1.819583	188.165.2.137	51.17.17.204	HTTP	1995	HTTP/1.1 200 OK (text/html)
44	1.820091	51.17.17.204	66.66.207.207	HTTP	59	HTTP/1.1 200 OK (text/html)

FIGURE VI.4.4. – Cette capture montre deux connexions distinctes

Cela présente un énorme avantage pour un réseau d'entreprise : on peut interdire aux utilisateurs d'accéder en direct à Internet, [au moyen d'un pare-feu](#) par exemple, et décider que toute communication vers l'extérieur doit passer par un proxy. Cela permet de contrôler qui accède à quoi et d'empêcher l'accès à certains serveurs ou sites web.

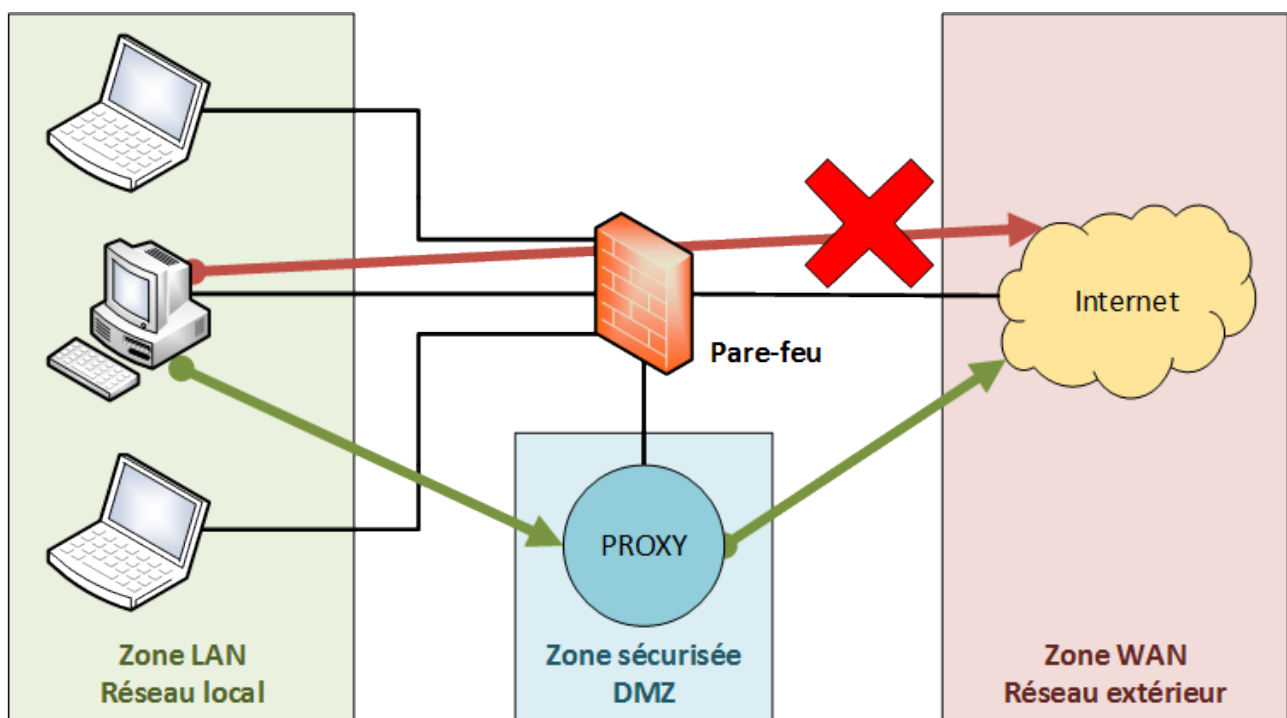


FIGURE VI.4.5. – Un pare-feu empêche l'accès direct à Internet

Mais ce fonctionnement a une autre implication. De nos jours, la plupart des communications sont chiffrées de bout-en-bout. Mieux : pour les protocoles les plus courants, elles le sont avec

**TLS.** Ce protocole, que nous avons survolé en fin de partie 5, chiffre l'intégralité des données utiles d'une manière telle qu'il est impossible de s'interposer, de déchiffrer le contenu, puis de le réexpédier comme si de rien n'était.<sup>1</sup> Pas moyen de filtrer ce qui passe... Ni même de savoir quel protocole applicatif est réellement utilisé ! Tout doit être transmis tel quel. Retenez ça, nous y reviendrons bientôt. 🍊

### VI.4.3. Limites et contournement

Vouloir contrôler ce que font ses employés ou ses élèves, ça peut se comprendre : on ne veut pas que les salariés ne jouent ou que les jeunes ne consultent des contenus classés X... Mais tout système peut être contourné par qui le veut vraiment, plus ou moins facilement. 🍊

Le filtrage des sites web est probablement le plus facile à flouer. Il suffit d'utiliser... un **webproxy** ! Il s'agit de sites web permettant de faire le relai vers un autre site. Ça fonctionne plus ou moins bien selon les fournisseurs et les sites visités. Cherchez "webproxy" sur un moteur de recherche, vous trouverez facilement. Et comme cette technique est répandue, la plupart des proxys en entreprise filtrent les webproxys ! 🍊

Alors pour trouver un webproxy qui ne soit pas bloqué, le mieux est encore d'utiliser le sien ! Il existe des solutions faciles à mettre en place, pour peu qu'on ait un peu de connaissances en serveur web, [comme phproxy](#) 🍊 . 🍊



FIGURE VI.4.6. – Navigation au travers d'un webproxy

Passons maintenant au niveau supérieur. Plus tôt, nous avons dit qu'il est impossible de différencier les protocoles se trouvant au dessus de TLS. Techniquement, on ne peut pas faire la différence entre HTTPS, SFTP ou **SSH**, qui fonctionnent tous sur TLS, à part peut-être

1. Pour que cela soit techniquement possible, il faudrait que le proxy dispose d'un certificat de chiffrement, qu'il soit explicitement reconnu par le client et, dans le cas du web, que le serveur ne s'y oppose pas. [Voir cet article sur HSTS](#) .

## VI. Reprenons du service

avec le numéro de port. Mais ce paramètre est changeable : sur son propre serveur, rien ne nous empêche de faire écouter un serveur web sur le port 1234 si ça nous chante. Une technique courante pour passer outre certains filtrages consiste à faire écouter un serveur **SSH** sur le port 443. Ce protocole sert principalement à contrôler un serveur à distance et il écoute normalement le port 22. Or, la quasi totalité des serveurs web utilisent le port 443, qui est le port standard pour HTTPS. On peut tout à fait établir une connexion **SSH** sur le port 443 : on n'y verra que du feu !

?

Ok, on a donc la possibilité de contrôler un serveur à distance discrètement... Tout ça pour ça ?

Oh non, ça va bien plus loin. 🍊 À partir du moment où vous pouvez ouvrir une connexion **SSH** vers votre serveur, vous avez la possibilité de faire passer n'importe quelle communication par cette voie. **SSH** permet d'ouvrir des tunnels dynamiques entre un client et un serveur : en ouvrant un port, le client devient alors à son tour... un proxy. Il transmet tout au serveur **SSH**, qui sert alors de... Oui, vous avez deviné : de proxy. 🍊 L'image suivante permet d'illustrer ce concept. On considère que le serveur **SSH** écoute le port 22, que le client écoute le port 12000, que le proxy écoute le port 8080 et que l'hôte X veut faire parvenir au serveur Y le message "Salut !" en TCP sur le port 11111.

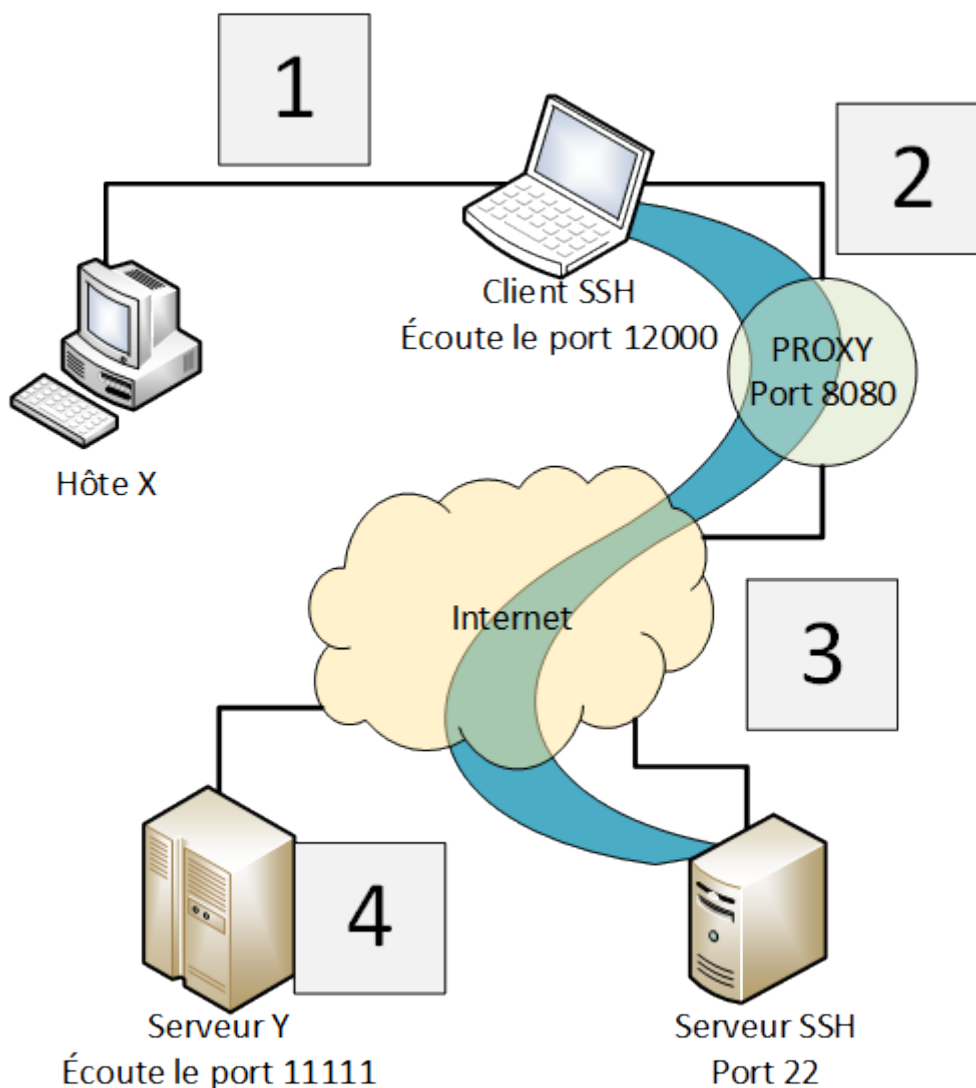


FIGURE VI.4.7. – Le tunnel SSH, en bleu, peut faire transiter n’importe quoi

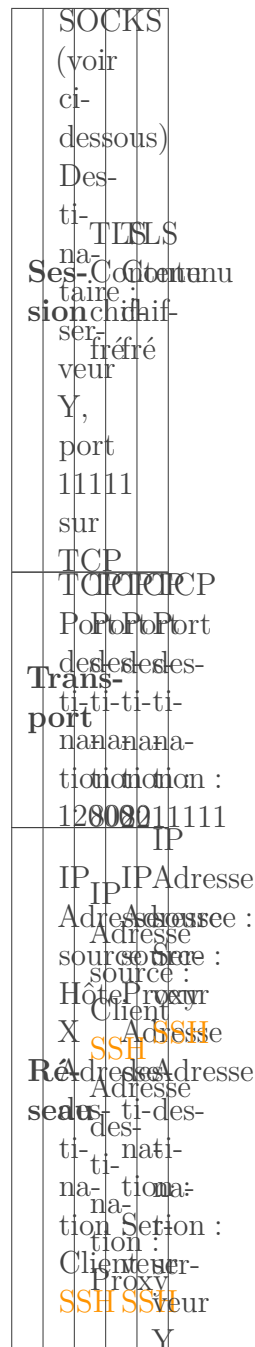
*i*


### Configuration du client

Tous les clients **SSH** ne permettent pas forcément l’ouverture de tunnels dynamiques. Voyez l’étape 4 de cet article [pour réaliser cela avec PuTTY](#), client en interface graphique très répandu, ou la section “Dynamic **SSH** Port Forwarding” de ce document [pour le client \*\*SSH\*\* Linux en ligne de commande](#).

Dans le détail, sur le réseau, on a les échanges suivants.

Étape	Client	Proxy	Serveur
1	Client SSH	PROXY	Serveur SSH
2	Client SSH	PROXY	Serveur SSH
3	Client SSH	PROXY	Serveur SSH
4	Client SSH	PROXY	Serveur SSH



En illustrant cette technique avec un message lambda ("Salut!"), on montre que presque n'importe quoi peut être transporté de cette manière. En fait, la communication entre l'hôte X et le client **SSH** se fait avec le **protocole SOCKS** , qui permet d'encapsuler n'importe quel protocole qui fonctionne sur TCP ou UDP. On peut donc faire transiter dans un tunnel **SSH** des pages web, des e-mails, des messageries instantanées, etc.

Sur l'image, nous avons représenté un hôte X, mais le poste ouvrant le tunnel peut tout à fait s'utiliser lui-même. Sur le même ordinateur, on peut avoir le client **SSH** et le navigateur web, par exemple, qui passe par le proxy créé par le tunnel. Dans ce cas, on configurera le navigateur avec pour adresse de proxy... l'adresse de loopback ! Voilà enfin un intérêt à ces fameux 127.0.0.1 et ::1 ! 🍊

On a abouti à une solution technique de contournement de filtrage plutôt pas mal, mais il reste un dernier obstacle qu'on rencontre couramment en entreprise : l'**authentification**. En

## VI. Reprenons du service

effet, n'importe qui ne peut pas utiliser les services d'un proxy : seul le personnel autorisé doit pouvoir s'en servir. Une méthode d'authentification classique dans les réseaux Microsoft est **NTLM**. Normalement, on ne parle pas de solution spécifique à un éditeur dans ce cours, mais dans le cadre de ce chapitre, nous allons faire une exception pour le plaisir de vous montrer une architecture qui vaut le détour. 🍌

**NTLM** est un système qui permet, entre autres, de s'authentifier auprès de proxys dans un environnement Microsoft. Si la plupart des clients **SSH** sont capables de passer par un proxy, peu supportent l'authentification **NTLM**. Pour pallier ce manque, on peut utiliser **CNTLM** [↗](#). Ce programme nécessite un peu de configuration, nous vous renvoyons à [ce billet \(en français\)](#) [↗](#) pour cela. Nous ne nous focaliserons que sur l'aspect réseau.

**CNTLM** se charge d'établir une connexion avec un proxy nécessitant l'authentification **NTLM**. Il ouvre un port et devient à son tour... un proxy, oui, on en aurait presque marre maintenant. 🍌 En faisant passer nos communications au travers de ce port, **CNTLM** se charge de les renvoyer au proxy avec l'authentification adéquate. On peut donc faire passer une connexion **SSH** par ce moyen, tout en établissant un tunnel, ce qui permet à n'importe quel logiciel de passer par le-dit tunnel... et faire transiter toutes ses communications par un serveur distant, au nez et à la barbe de votre méchant patron qui ne veut pas que vous jouiez à des jeux en ligne au bureau ! 🐱



On plaisante, mais gardez à l'esprit qu'une telle pratique est interdite dans la plupart des entreprises et établissements d'enseignement. Bien que ce ne soit pas en soi illégal, contourner des mesures de sécurité peut vous valoir une sanction disciplinaire. Autrement dit, vous n'irez pas en prison pour ça mais vous pouvez vous faire virer.

Voici une illustration de la chaîne de communications qui s'établit si vous visitez Zeste de Savoir au travers d'une telle solution. Le client **SSH** écoute sur le port 12000 pour son tunnel, **CNTLM** écoute sur le port 6666, ces valeurs sont arbitraires. Le serveur **SSH** écoute sur le port 443 pour éviter un éventuel filtrage. Le proxy écoute sur le port 8080, c'est une valeur courante dans la configuration des proxys. Quant à Zeste de Savoir, il écoute sur le port 443, qui est le port standard des serveurs web.

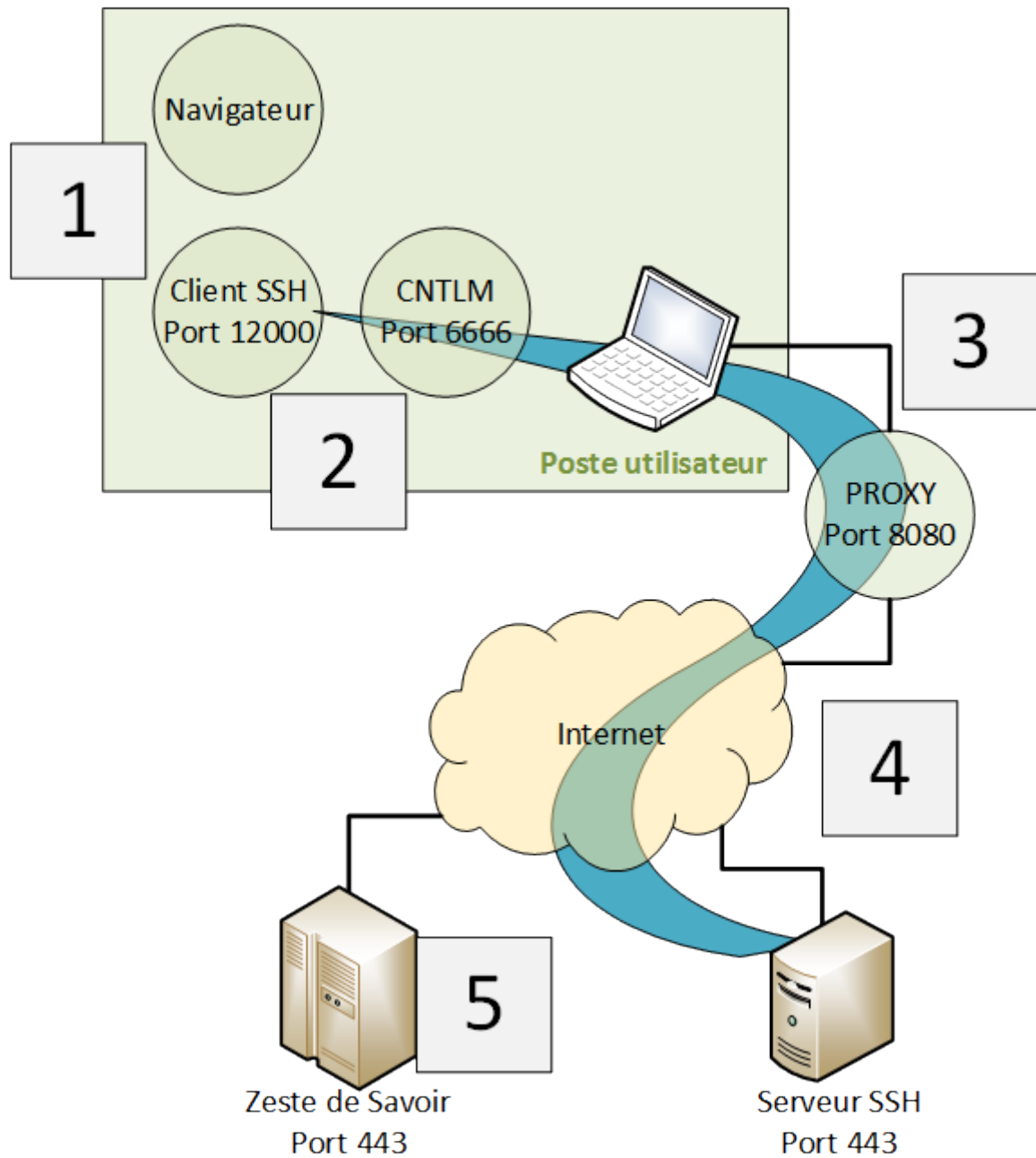


FIGURE VI.4.8. – Imbrication de proxys pour atteindre un serveur distant

C'est à dire  
 (OS) 3 4 5  
 Re- Connexion  
 quête SSH  
 HTTP HTTPS  
 Ap- vers proxys  
 pli- Connexion  
 ca- zes- pense-  
 tion te- que-  
 de- c'est-  
 sa- dudusa-  
 voir. voir. Som



Cette architecture est un petit exploit technique qui illustre un proverbe du monde de la sécurité : hacker vaillant, rien d'impossible ! 🍌

## Conclusion

Un proxy, ça semble être quelque chose de simple à première vue : un relai qui sert de filtre. Mais quand on creuse, on se rend vite compte que ce service soulève plein de problématiques liées à la sécurité des réseaux. C'est justement le thème de la partie suivante.

## VI.5. Dans le collimateur de la supervision

### Introduction

Dans l'univers des réseaux, il existe un super-héros doté d'une vue perçante, capable d'apercevoir la moindre défaillance même sur les appareils les plus isolés. On l'appelle... Super-vision !



FIGURE VI.5.1. – Et on remercie l'intelligence artificielle Dall-E pour cette création incroyable. Bon, en réalité, on oublie le slip rouge et les décors de bande dessinée. Dans les faits, la

**supervision** est un concept permettant de s'assurer que tout fonctionne correctement dans un réseau. Pourquoi est-ce important ? Comment la met-on en place et quels protocoles se cachent derrière ? C'est ce que nous allons voir dans ce chapitre.

### VI.5.1. Intérêt en cas de dommage

Tout réseau informatique permet à des utilisateurs d'accéder à des services, que ce soit pour le loisir, le boulot, un besoin quelconque, etc. Que se passe-t-il si, soudainement, vous ne pouvez plus travailler en raison d'une panne d'un switch dans votre bureau ?

?

Eh bien je vais prendre un café et je discute avec les collègues le temps que ce soit résolu, quelle question !

Cela sous-entend que le problème doit être résolu. Et donc que les personnes en charge de cette résolution soient au courant. Si vous en profitez pour prendre du bon temps sans rien dire à personne, vous mettez du monde dans l'embarras : vos collègues ne peuvent pas travailler, donc l'entreprise perd de l'argent, potentiellement vos clients sont impactés aussi puisque vous ne pouvez rien pour eux pendant ce temps, et ainsi de suite. Heureusement, il existe des solutions ~~contre les fainéants comme vous~~ pour alerter immédiatement les responsables des infrastructures en cas de défaillance. C'est le principe de la supervision.

Avant de plonger dans l'aspect purement technique du sujet, précisons que la supervision ne sert pas uniquement à repérer les équipements en panne. Elle permet d'alerter sur un certain nombre d'anomalies, comme un câble débranché ou branché de manière inhabituelle, une variation anormale du temps de réponse d'un équipement, un accès réseau défaillant, etc. Cette sécurité est vitale dans les entreprises qui fournissent un service ayant besoin d'être accessible en permanence. Bien évidemment, cela représente un coût : il faut bien payer les gens qui vont devoir intervenir en urgence si une alerte se déclenche. C'est comme pour les assurances : se passer de ce service est un risque qu'il vous appartient d'évaluer.

Maintenant, si c'est vous qui êtes responsable de la surveillance des infrastructures, il vous sera utile de savoir comment la supervision fonctionne et comment la mettre en place.

### VI.5.2. Comment ça marche ?

Prenons le cas d'une infrastructure d'entreprise qui compte de nombreux sites, qui eux-mêmes disposent de plusieurs équipements réseaux comme des commutateurs et des routeurs, mais aussi des serveurs, des imprimantes, etc. On souhaite pouvoir être proactif en cas de panne, c'est-à-dire intervenir dès qu'une défaillance est détectée informatiquement, avant même que les gens qui travaillent ne se mettent à râler. Une solution simple consiste à disposer d'un élément capable de joindre tous les éléments sur le réseau et émettre une alerte si l'un d'eux ne répond plus. Dit plus prosaïquement, une machine [pingue](#) ☞ régulièrement tout le monde. Facile et efficace... dans une certaine mesure. En se reposant juste sur ICMP, impossible d'avoir des informations détaillées comme le taux de saturation d'un équipement, quel lien est dysfonctionnel, si une route a été modifiée, etc. Pour permettre d'intervenir efficacement en disposant de toutes les informations utiles, on recourt à un protocole spécifique : SNMP.

SNMP, pour Simple Network Management Protocol, fonctionne sur UDP et utilise par défaut le port 161. Il est pris en charge par la plupart des systèmes d'exploitation et des constructeurs de

## VI. Reprenons du service

matériel réseau : Cisco, Fortinet, Juniper, etc. Sa mise en place est relativement simple, il suffit d'activer le service de supervision, que l'on appelle **agent**, généralement au moyen d'une ligne de commande, en spécifiant trois paramètres principaux :

- l'adresse du **superviseur**, ou *manager*, qui est l'équipement chargé de superviser l'infrastructure ;
- la version du protocole SNMP, sachant que les versions 1 et 2 n'utilisent pas de chiffrement, il convient d'utiliser la version 3 pour un minimum de sécurité ;
- les communautés.

Ce dernier point est particulier. Une communauté désigne un ensemble d'équipements qui sont destinés à être supervisés par le même manager. Le nom de la communauté doit rester confidentiel, car il agit comme un mot de passe. Une personne qui connaîtrait le nom de la communauté SNMP d'un équipement pourrait communiquer avec lui pour obtenir des informations précises, voire pire. En effet, ce protocole permet de récupérer des pans entiers de configuration, le contenu de tables de routage, etc.

?

Au pire, c'est pas catastrophique, tant qu'on ne récupère pas de mots de passe...

C'est sans compter sur le fait qu'on peut définir deux communautés sur chaque équipement : une *read-only* (RO), qui ne permet au superviseur que d'obtenir des informations, et une *read-write* (RW), qui lui permet également de **modifier la configuration** ! Une tierce personne qui obtiendrait un accès à une communauté RW pourrait mettre un sacré bazar dans toute l'infrastructure.

Les agents SNMP peuvent communiquer avec leur superviseur de deux manières :

- requête à l'initiative du manager
- envoi d'une alerte au manager de la part des agents.

Dans le premier cas, c'est simplement que le superviseur souhaite obtenir une information (nom de la machine, statut des interfaces, ...) ou changer la configuration, dans le cas d'une communauté RW. Une requête **GET** ("obtenir") ou **SET** ("définir") est simplement envoyée à l'agent visé, qui lui répond. On appelle cela du **polling**. La structure des éléments pouvant être traités est standardisée et constitue la **MIB** (*Management Information Base*, base d'informations de gestion). Chacun de ces éléments porte un numéro spécifique appelé **OID** (*Object Identifier*). Les outils de supervision leur donnent des noms plus explicites, comme on peut le voir sur la capture d'écran suivante.

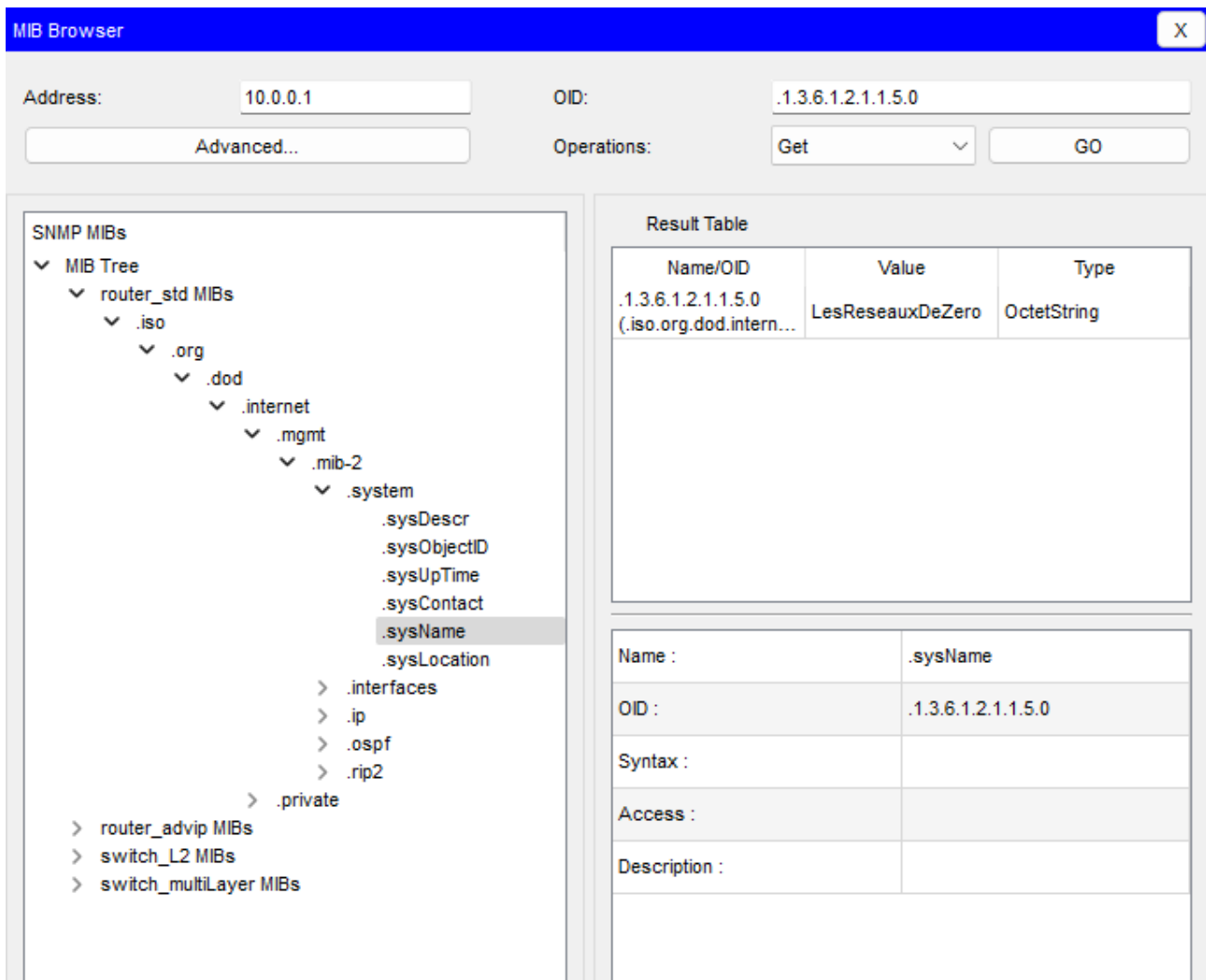


FIGURE VI.5.2. – L’explorateur de MIB, sur le logiciel Cisco Packer Tracer



L’agent ne répond pas si le nom de la communauté est incorrect.

Si un évènement survient, comme le débranchement d’un câble, une perte de paquets ou une panne quelconque, les agents SNMP peuvent envoyer une alerte sous forme de **message trap**. Ces messages peuvent contenir plusieurs informations telles que le niveau de gravité, l’OID concerné, ou encore des informations complémentaires définies dans la configuration. Par défaut, les traps SNMP sont envoyés au superviseur en UDP sur le port 162.

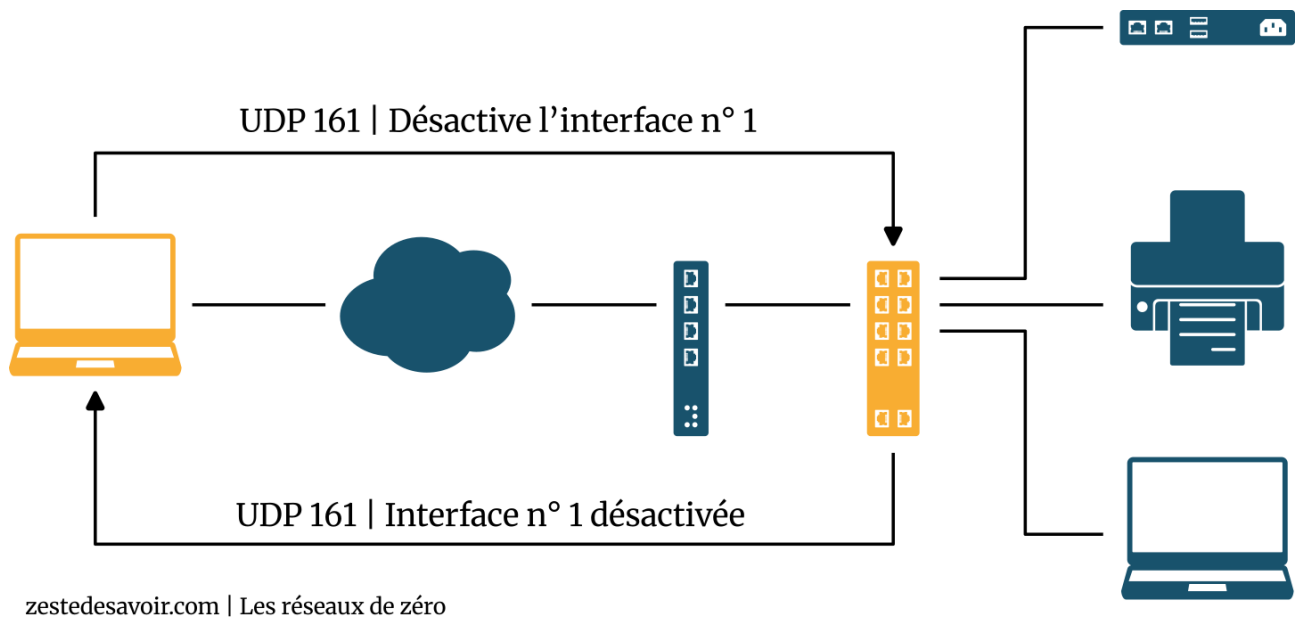


FIGURE VI.5.3. – Le polling SNMP consiste à contacter les agents pour leur demander une information ou envoyer une instruction (CC BY)

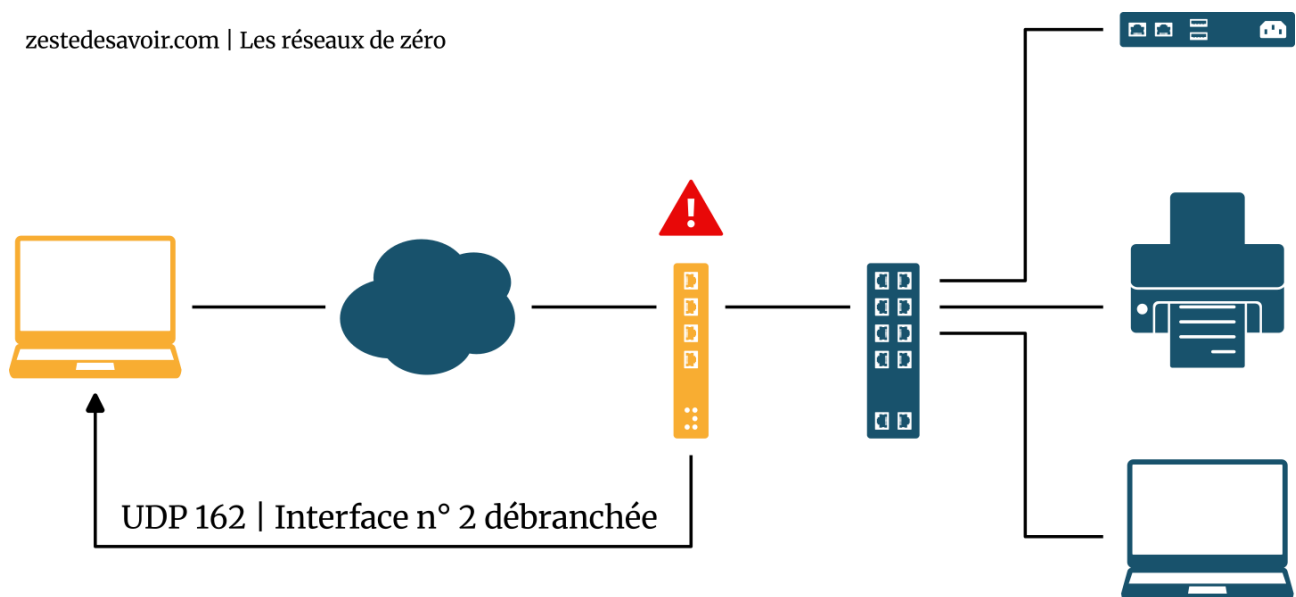


FIGURE VI.5.4. – Le trap SNMP permet à un agent d'informer immédiatement le superviseur d'un évènement (CC BY)

i

Avec SNMP, on peut superviser des équipements depuis n'importe où, sans être physiquement sur place. Ainsi, il est courant pour de grandes entreprises ou administrations d'avoir des équipes concentrées dans un seul endroit, qui surveillent des milliers de sites répartis partout dans le monde, prennent la main à distance sur les équipements quand c'est possible et appellent des personnes sur place quand une intervention physique est nécessaire.

### VI.5.3. Exemples de superviseurs

Explorer des MIB manuellement, ce n'est ni passionnant, ni efficace. Plusieurs logiciels permettent d'automatiser la supervision et de générer des graphiques, des représentations visuelles de l'infrastructure, etc. Passons en revue plusieurs d'entre eux.

#### VI.5.3.1. Zabbix

[Zabbix](#) est un logiciel libre fournissant une interface web pour présenter des informations sur les machines supervisées. Il est possible de programmer des actions à réaliser lorsqu'un évènement se produit, comme par exemple envoyer un e-mail lorsqu'une machine ne répond plus aux sollicitations. En plus de SNMP et ICMP, Zabbix gère d'autres protocoles comme HTTP pour vérifier le fonctionnement de services web. Davantage de services peuvent être monitorés avec l'installation d'un agent spécifique sur les machines surveillées, qui n'est pas nativement présent sur les systèmes usuels.

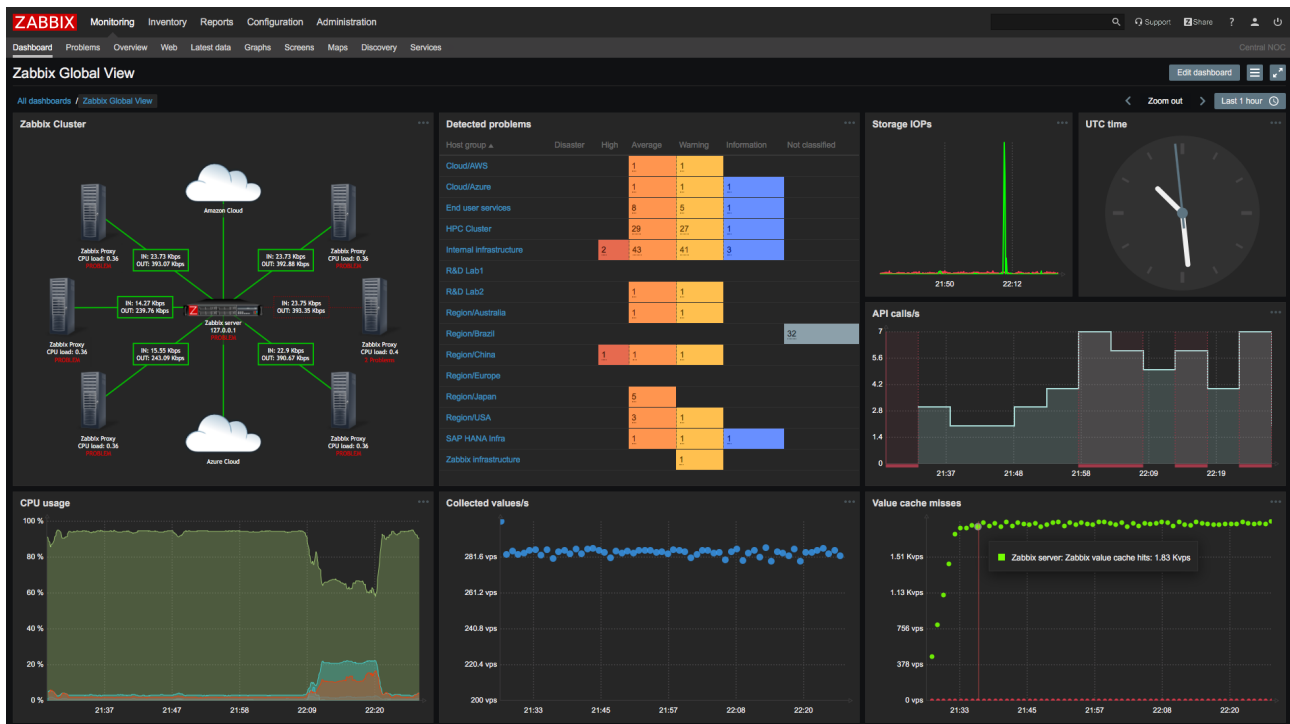


FIGURE VI.5.5. – Un tableau de bord Zabbix. Image par Dotneft distribuée sous licence CC BY SA 4.0. Source : [https://commons.wikimedia.org/wiki/File:Dashboard\\_graphs\\_v4\\_dark\\_1.png](https://commons.wikimedia.org/wiki/File:Dashboard_graphs_v4_dark_1.png)

#### VI.5.3.2. DX Spectrum

[DX Spectrum](#) est un programme commercial édité par la société Broadcom. Il ne convient qu'aux sociétés ayant une grosse infrastructure à superviser, car le logiciel ne peut pas être acheté directement, il nécessite un partenariat avec son éditeur. L'interface permet un aperçu schématique du réseau avec un code couleur plutôt intuitif : les équipements représentés en vert ne présentent aucun problème, ceux en orange nécessitent une attention (service dysfonctionnel, pertes de paquets, ...), ceux en rouge présentent un problème grave. Les fonctionnalités intégrées à ce logiciel incluent tout ce qu'on peut attendre d'un outil de supervision.

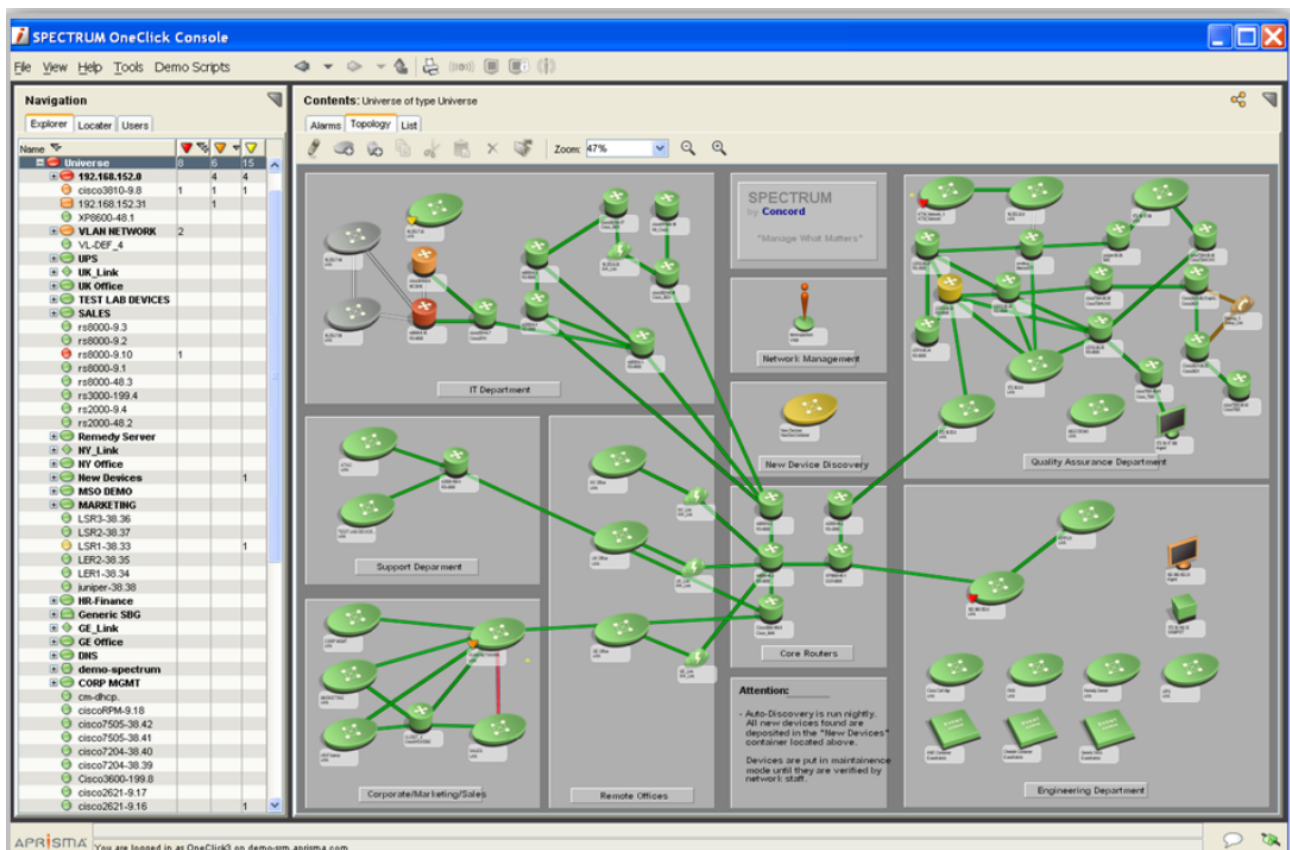


FIGURE VI.5.6. – Capture d’écran de DX Spectrum issue d’une documentation de la société Broadcom

### VI.5.3.3. Nagios

À l’instar de Zabbix, [Nagios](#) est une solution libre pour la supervision, qui propose une interface web permettant de visualiser les informations que l’on souhaite. Il offre la possibilité d’intégrer de nombreux *plugins* pour monitorer tout type de service et de réaliser des actions en conséquence (démarrage à distance d’un serveur de secours en cas de défaillance d’un équipement donné, ...).

## VI. Reprenons du service

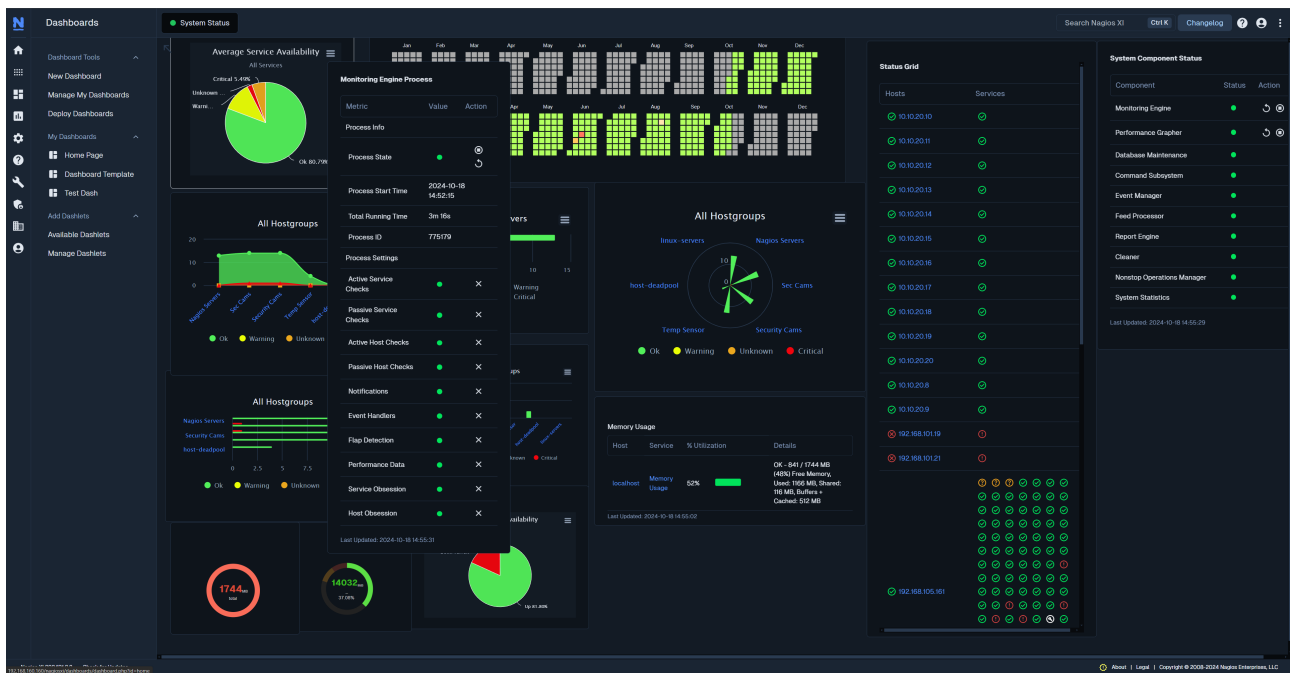


FIGURE VI.5.7. – Capture d'écran de Nagios provenant du site web de l'éditeur

Bien d'autres logiciels existent pour assurer la supervision d'infrastructures réseaux. Nous n'en avons présenté que quelques-uns que nous avons déjà utilisés dans des contextes professionnels. Il ne s'agit en aucun cas de recommandations.

## Conclusion

La supervision est un service qui dépend davantage des moyens que l'on souhaite investir que de contraintes techniques. Il existe de nombreuses solutions de monitoring ; l'exploitation et les commandes sont différentes en fonction des systèmes et des constructeurs. Les bases que nous avons étudiées dans ce chapitre vous permettront d'appréhender plus facilement les programmes que vous rencontrerez en milieu professionnel.

## Conclusion

Heureusement que des gens (parfois tordus, il faut bien le dire 🍊 ) ont eu l'idée d'inventer ces services ! Sans eux, utiliser un ordinateur ou une tablette serait inenvisageable pour le commun des mortels. 🍊

# Septième partie

## Évolutions

# Introduction

Si les principes de base du réseau n'ont que peu évolué fondamentalement en quelques décennies, l'usage qui en est fait a drastiquement changé depuis le XX<sup>e</sup> siècle. Les années 2010 et 2020 ont vu le développement de technologies nouvelles sur lesquelles reposent nos services modernes. Dans cette partie, nous allons découvrir des principes récents tels que le cloud ou le **SDN**.

## VII.1. Saint Cloud, priez pour nous

### Introduction

Parmi les termes à la mode, celui de cloud revient fréquemment. On l'emploie sans trop comprendre de quoi il s'agit, même dans des médias grand public. Dans ce chapitre, nous allons clairement définir ce concept, revenir sur son histoire, comprendre son intérêt technique et quels en sont les principaux fournisseurs de service.

#### VII.1.1. Une pluie de services

Si vous demandez à des profanes ce qu'est le cloud, il y a des chances qu'on vous réponde qu'il s'agit d'un moyen de stocker ses données pour y accéder de partout. Cet usage est le plus connu du grand public, mais ce n'est qu'une petite partie de ce dont il s'agit réellement. Initialement, le cloud, ou en bon français **informatique en nuage** (sérieusement, qui emploie cette expression ?), désigne le fait d'accéder à un service exécuté sur une machine distante. Ce concept vous paraît récent ? Les années 2010, peut-être fin des années 2000 ? Faites chauffer votre Tardis, on considère que cela remonte aux années 1950 ! 🍌 En réalité, à cette époque, les ordinateurs occupaient des pièces entières et on pouvait y connecter des terminaux pour utiliser des services qui s'exécutaient sur ces mastodontes. Techniquement, on rentre bien dans la définition évoquée précédemment.

Mais ~~dans l'antiquité~~ au milieu du XX<sup>e</sup> siècle, on ne parlait pas encore de cloud. Le terme s'est imposé de lui-même plus tard, à partir des années 1990, et viendrait du fait que les architectes dessinaient des nuages sur des schémas pour représenter des réseaux étendus comme Internet. De nos jours, les services qu'on retrouve dans le cloud peuvent inclure tout type d'application, mais aussi de la mise à disposition d'infrastructure, comme nous allons le voir par la suite.

Une autre façon de voir les choses, c'est de considérer que le cloud, c'est la consommation de puissance de calcul. De la même manière qu'on peut consommer de l'électricité, du gaz ou de l'eau, le cloud offre la possibilité d'ouvrir un robinet délivrant de la puissance de calcul à la demande. Cette dernière peut être utilisée pour fournir des applications, réaliser des opérations complexes, mettre à disposition des machines virtuelles, etc. Notez que lorsque vous ouvrez votre robinet d'eau, vous ne savez pas forcément de quelle source elle provient ni quels équipements l'ont filtrée ou purifiée. De même, lorsque vous allumez la lumière, vous ne savez pas si l'énergie électrique qui fait briller votre lampe est d'origine fossile, nucléaire, renouvelable, ni de quelle centrale elle provient. Pour le cloud, le principe est comparable : **les clients et utilisateurs ne savent pas quelles machines ils sollicitent**. Dans une architecture client-serveur traditionnelle, vous contactez un serveur, qui vous répond. Vous savez qui vous contactez et où sont envoyées les données que vous transmettez. Avec le cloud, ce n'est plus le cas : personne ne sait à l'avance quelle machine va traiter vos requêtes. Fondamentalement, pour les clients, rien ne change : on envoie toujours des paquets IP dans lesquels on encapsule d'autres protocoles, comme vu tout au long du cours. Mais côté serveur, l'infrastructure cloud va, en fonction de paramètres comme la latence ou la charge, renvoyer les échanges vers une machine, le plus souvent virtuelle. Et peu

importe laquelle : toutes sont configurées de la même manière, toutes vous répondront la même chose.

On distingue trois grandes catégories de modèles de cloud : **IaaS**, **PaaS** et **SaaS**.

### VII.1.1.1. Infrastructure

Un cas assez simple à appréhender est l'**Infrastructure as a Service**. Ce principe consiste à fournir des instances de systèmes d'exploitation, des machines virtuelles, des espaces de stockage, des réseaux, bref, tout ce qui constitue l'infrastructure informatique d'une entreprise. Les clients vont être principalement des administrateurs systèmes et réseaux, ou des ingénieurs. Avec ce modèle, les applications à destination d'utilisateurs finaux, comme les e-mails ou la visioconférence, ne sont pas fournis. Les clients font ce qu'ils veulent de l'infrastructure qu'ils commandent, mais c'est à eux de gérer toute la partie logicielle.

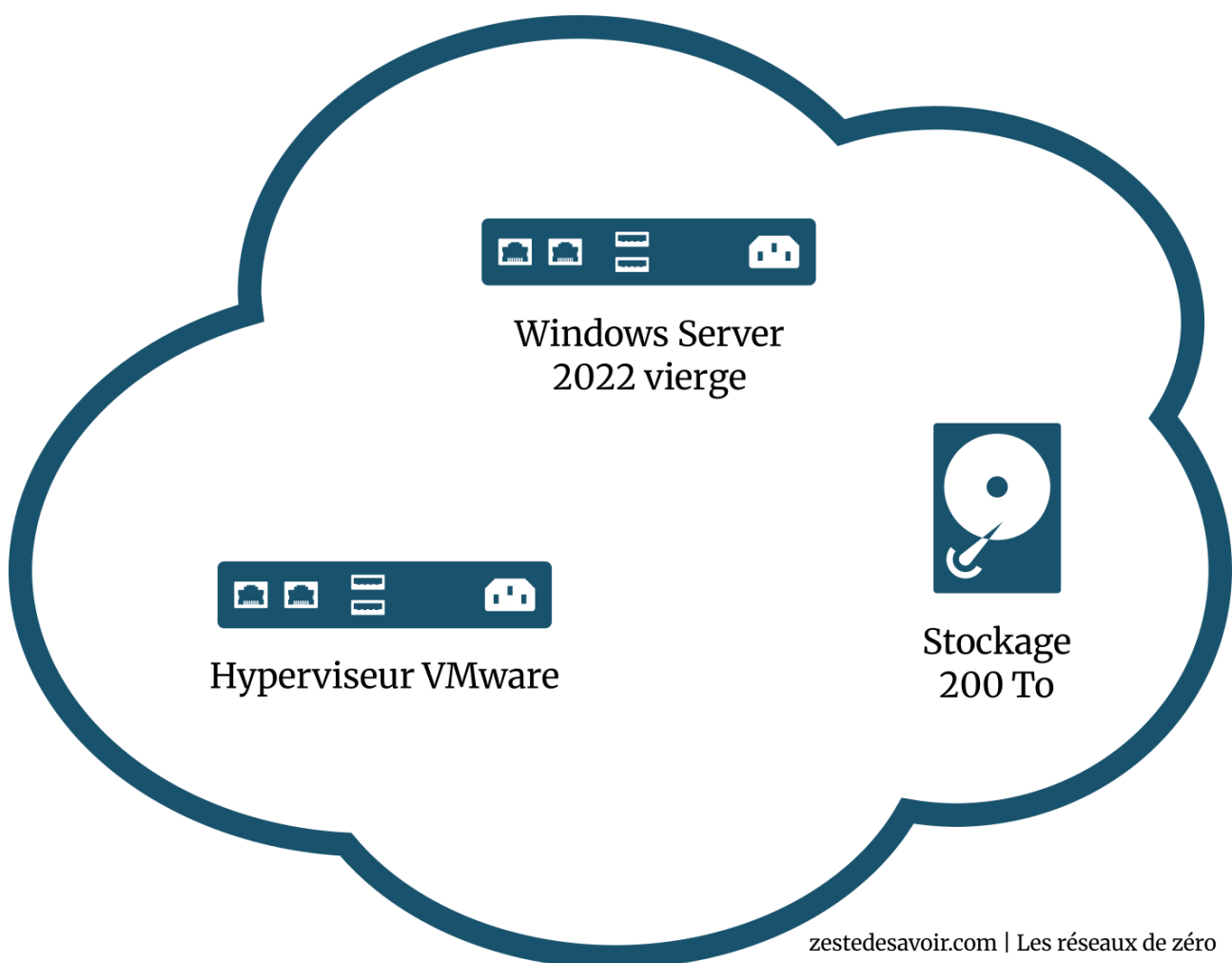


FIGURE VII.1.1. – Infrastructure as a Service : l'infrastructure fournie n'inclut pas les logiciels (CC BY)

### VII.1.1.2. Plateforme et fonction

Le modèle **Platform as a Service** ajoute, en plus de l'infrastructure, des outils de base pour programmer des applications, comme des environnements de développement, des bibliothèques

logicielles, des bases de données, etc. Il est utilisé principalement par des développeurs, qui n'ont ainsi pas à se soucier de l'installation et la configuration de serveurs.

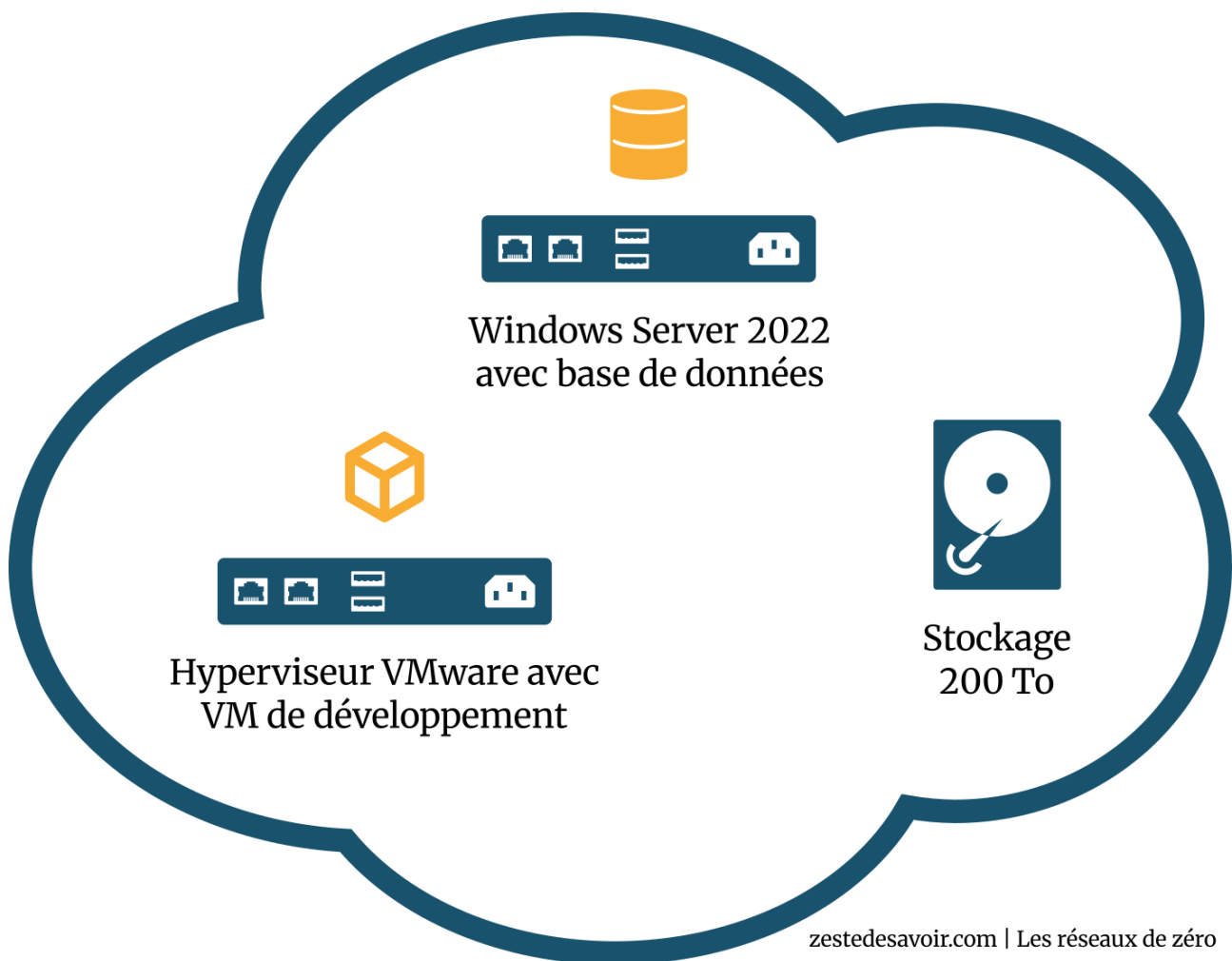


FIGURE VII.1.2. – Platform as a Service : l'infrastructure inclut des outils de développement (CC BY)

Une variante de ce modèle est le **Function as a Service**. Principalement utilisé par les développeurs aussi, il consiste à programmer des fonctions qui interagissent les unes avec les autres et qui se déclenchent uniquement quand un évènement donné survient. Ainsi, dans ce modèle, une infrastructure cloud **FaaS** fera exécuter une fonction, et non un programme entier, à un serveur. Cela permet notamment de répartir plus finement la charge de travail des machines. Cela rentre dans le concept *serverless* ("sans serveur"), qui signifie non pas qu'on se passe de serveur, mais que les développeurs n'ont pas à se soucier du serveur sur lequel leur code est exécuté (pas de maintenance, de gestion de la capacité, etc. : tout cela est géré par le fournisseur).

### VII.1.1.3. Logiciel

Le **Software as a Service** est le modèle de cloud le plus proche des utilisateurs finaux, puisqu'il consiste à fournir des applications en ligne à (potentiellement) tout le monde. En fait, on y a recours en permanence ! Des services comme Netflix, OpenStreetMap ou GMail reposent sur

ce modèle : on n'a rien sur nos ordinateurs, smartphones ou tablettes, on accède à nos séries, cartes et messages par Internet où tout est stocké et calculé chez des fournisseurs. Tout au plus, on dispose d'une application pour y accéder, et encore, on peut simplement utiliser notre navigateur web.

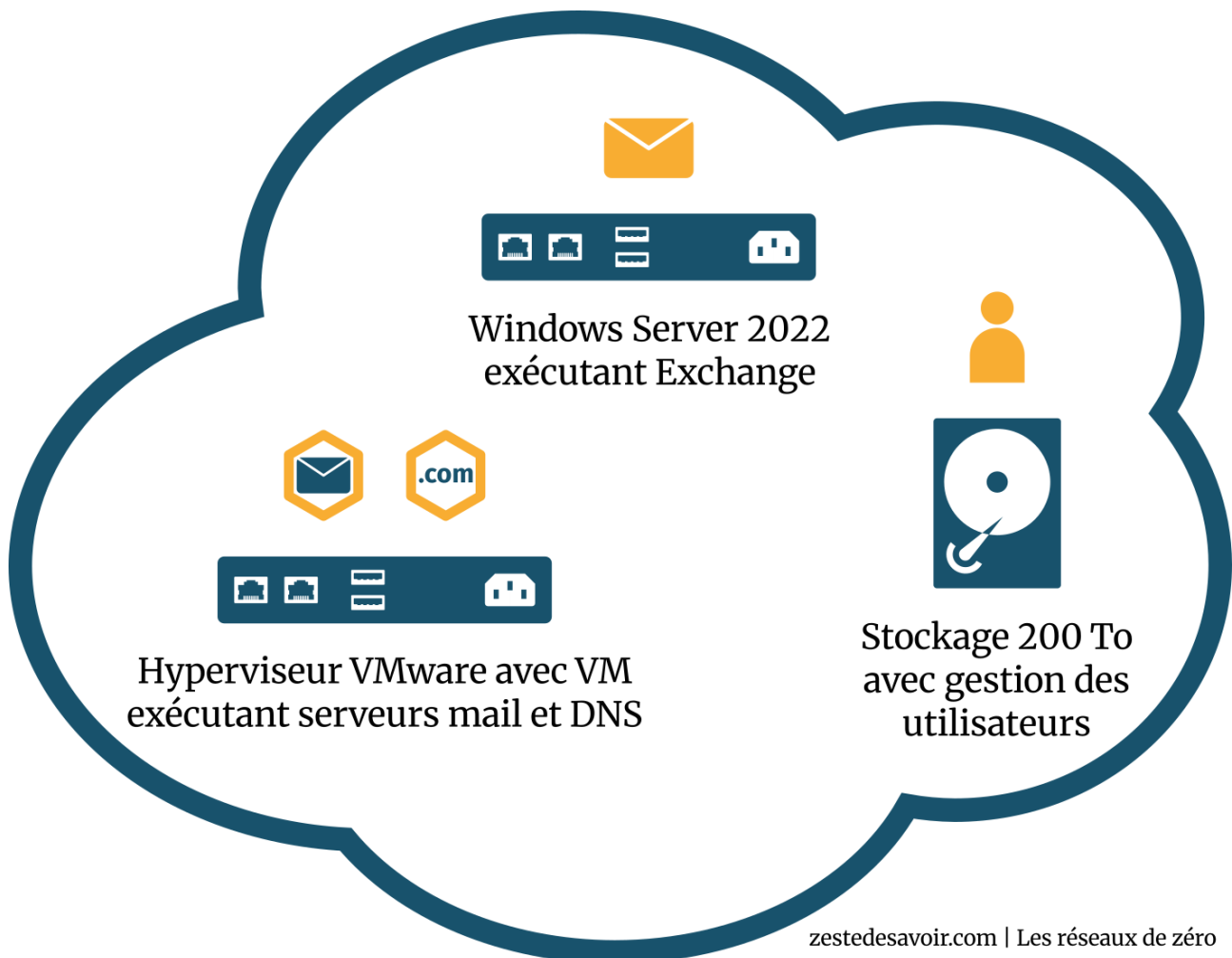


FIGURE VII.1.3. – Software as a Service : les applications sont fournies par le prestataire de cloud (CC BY)

### VII.1.1.4. Stratégie

Nous avons comparé le cloud à des réseaux publics comme l'électricité ou le gaz. Lorsque des services cloud sont accessibles à tout type de client, on parle aussi de **cloud public**. Par exemple, OneDrive est un service de cloud public proposé par Microsoft pour le stockage de données, et peut être utilisé par toute personne qui le souhaite. On peut aussi citer ses concurrents iCloud d'Apple ou encore Dropbox.



La notion de cloud public signifie uniquement que n'importe qui<sup>1</sup> peut profiter de ces services. Cela n'implique en aucun cas le fait que ces derniers soient gratuits ou détenus par un État ou une collectivité.

*A contrario*, lorsqu'une infrastructure cloud est interne à une entreprise, et utilisée uniquement pour ses propres besoins, on parle de **cloud privé**. De nombreuses sociétés privées disposent de services de cloud qui ne servent que leurs propres besoins et dont personne d'autre ne peut bénéficier.

i

Il est possible de mixer cloud public et privé pour optimiser ses coûts ou pour des questions de sécurité des données. Dans ce cas, on parle de **cloud hybride**. On peut même recourir à plusieurs fournisseurs de cloud public en simultané, généralement pour répondre à des besoins différents et disposer des solutions les plus adaptées : le stockage de données chez Google, les services de messagerie chez Microsoft, etc.<sup>2</sup> On appelle cela le **multicloud**.

Ce qui justifie les choix en matière de cloud, qu'il s'agisse des stratégies d'hébergement, des modèles employés ou même la décision d'y recourir ou non, c'est en partie les coûts que représentent la réponse technique à des besoins. Concrètement et toutes choses égales par ailleurs, si vous devez choisir entre acheter un serveur à 5.000 € qui va durer 10 ans, ou payer 400 € par an pour un service équivalent avec le cloud, vous allez sûrement vous tourner vers cette dernière solution qui est plus avantageuse. Pour comprendre comment il est possible de proposer des offres compétitives, nous devons étudier le modèle économique du cloud.

### VII.1.2. Modèle économique

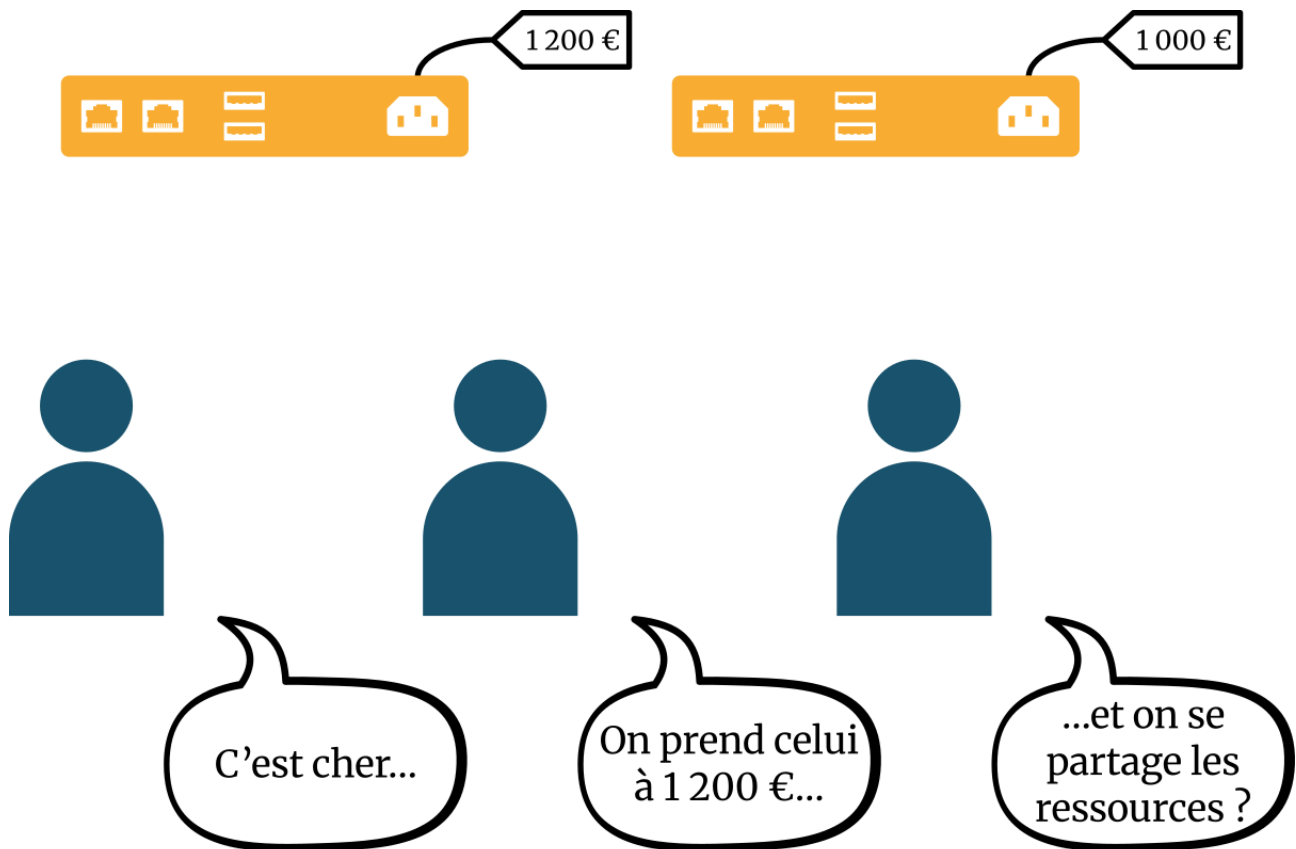
#### VII.1.2.1. La mutualisation

Le principe fondamental du cloud sur le plan économique est la **mutualisation**. Des ressources, par exemple des serveurs, sont mis en commun pour servir des clients différents et réduire le coût total. Supposons que, pour leurs sites web respectifs, Zeste de Savoir et les éditions Eyrolles veuillent chacun acquérir un serveur qui coûte 1000 €. Avec ce matériel, chacun est sûr de pouvoir absorber ses pointes de trafic. Mais en période creuse, ces appareils sont sous-exploités. En **mutualisant** les serveurs, c'est-à-dire en n'en utilisant qu'un pour deux, les coûts sont divisés par deux pour chaque entité ! Et même s'il faut envisager l'achat d'un serveur un petit peu plus cher pour assurer qu'il supporte les pics de trafic, cela fait tout de même une belle économie. C'est le principe de la mutualisation, qu'on retrouve dans de nombreux domaines de la vie courante : covoiturage, assurances, colocation, etc.

---

1. N'importe qui pour peu que la personne paie et respecte les conditions d'utilisation, bien entendu. Les sociétés privées peuvent généralement se réserver le droit de refuser des clients.

2. Il s'agit d'un exemple quelconque et non d'une recommandation.



zestedesavoir.com | Les réseaux de zéro

FIGURE VII.1.4. – Réaliser un achat en commun et se partager les ressources peut être avantageux (CC BY)

Le cloud pousse ce concept à l'extrême en offrant une grande souplesse pour ses clients. Cela se traduit par des capacités d'adaptation instantanées : on peut augmenter la puissance de calcul de son infrastructure en un clic, ou bien baisser la quantité de mémoire vive temporairement, ou encore absorber une consommation soudaine de ressources qui n'étaient pas prévues sans ralentir l'expérience des utilisateurs. C'est ce qu'on appelle la **scalabilité**<sup>1</sup> (ou en anglais *scalability*).



zestedesavoir.com | Les réseaux de zéro

FIGURE VII.1.5. – Le cloud permet d’ajuster facilement les ressources disponibles pour les utilisateurs (CC BY)

Pour fournir des services avec une telle capacité d’adaptation, de nombreuses ressources sont nécessaires, tant en puissance de calcul qu’en mémoire et qu’en bande passante. Le réseau est un point aussi important que le matériel : les échanges de données doivent être ultra rapides, que ce soit entre les serveurs ou avec les utilisateurs distants. Une telle infrastructure ne peut pas être déployée n’importe où.

### VII.1.2.2. Les datacentres

Le terme de **datacentre** désigne des espaces physiques aménagés de manière spécifique pour accueillir des infrastructures de type cloud. Vous avez peut-être déjà vu des images de ces rangées d’armoires remplies d’appareils comme des routeurs, des serveurs, des switches, des pare-feux, etc. qui semblent clignoter comme des sapins de Noël. Là aussi, les datacentres exploitent le principe de mutualisation : ils peuvent héberger de nombreux clients à la fois, y compris des fournisseurs de cloud. Les charges associées à un tel endroit sont colossales, alors autant les réduire autant que possible ! La climatisation, l’électricité, les systèmes de refroidissement, la sécurité sont autant de postes de dépense qu’il n’est pas envisageable de négliger.

/opt/zds/data/contents-public/les-reseaux-de-z

Image par Robert.Harker sous licence CC BY SA 3.0 [↗](#)

Ce dernier point est crucial et doit être envisagé sous toutes ses facettes. Les datacentres hébergent des quantités astronomiques de données sensibles qu’il faut protéger, aussi bien des intrus que d’éventuels accidents. Il n’est pas rare que l’accès physique à ces endroits nécessite plusieurs moyens [d’identification](#), [d’authentification](#) et [d’autorisation](#) [↗](#) : contrôle de documents

d'identité, analyse de données biométriques comme les empreintes digitales ou l'iris, ... Des mesures de surveillance sont généralement déployées, telles que des vigiles ou des caméras. La sécurité physique des personnes est aussi prise très au sérieux... Ce qui n'empêche pas des catastrophes comme [les incendies de mars 2021](#) qui ont en partie détruit les datacentres d'OVHCloud à Strasbourg, dans l'est de la France, à la frontière avec l'Allemagne. [Un rapport d'enquête](#) relève de nombreux manquements à la sécurité du matériel et des données dans cet accident.

De nos jours, dans les années 2020, la tendance est assurément au cloud. De nombreux appareils connectés n'ont pas d'intelligence embarquée, c'est-à-dire qu'ils ne sont pas capable de traiter les données qu'ils collectent eux-mêmes. Par exemple, certains postes de radio connectés ne conservent même pas en mémoire interne les adresses des flux enregistrés, ils interrogent le cloud pour savoir quoi faire au moindre appui sur un bouton. Ce qui fait que quand le fournisseur de service subit une panne momentanée, on ne peut plus utiliser l'appareil, même si les flux audio que l'on souhaite écouter sont bien disponibles ! Cela est très frustrant et oui, c'est du vécu. Plus prosaïquement, le matériel qui ne fait que générer des datas comme les capteurs se contente d'envoyer toutes ses données dans le cloud, y compris celles n'ayant aucun intérêt. On ne peut pas le lui reprocher, il a été conçu pour cela et ne dispose pas de capacité de traitement propre ! Pourtant, cela représente un certain gâchis. Transmettre des données a toujours un coût, on doit bien payer la bande passante utilisée, l'électricité que consomme l'infrastructure pour transmettre et traiter l'information, etc. Si on peut le réduire, autant le faire.



L'ensemble des appareils connectés, tels que les montres, les capteurs, les liseuses, etc. forme ce qu'on appelle l'**Internet des objets** ou IoT (*Internet of things*).

### VII.1.2.3. Sur le bord

Plutôt que de confier l'intégralité des datas générées à un service lointain, on peut tenter de décentraliser la fourniture de services. Si tous les abonnés européens à Netflix ou Prime Video doivent streamer leurs séries depuis des serveurs aux États-Unis, le coût en termes de trafic est plus élevé que si la source se trouve en plein centre de l'Europe, par exemple à Berlin. Eh bien faisons une copie de ces serveurs à Berlin, et synchronisons-la avec l'infrastructure d'origine ! C'est le principe du **Content Delivery Network** ou CDN. Ce concept permet de réaliser des économies en décentralisant des services gourmands en ressources, notamment la vidéo à la demande. Il est apparu dans les années 1990. On parle aussi d'*edge computing*. Cette expression imagée renvoie au fait que les clients accèdent aux services par l'intermédiaire d'infrastructures en bordure (*edge*) d'une infrastructure centrale, donc plus proches d'eux.

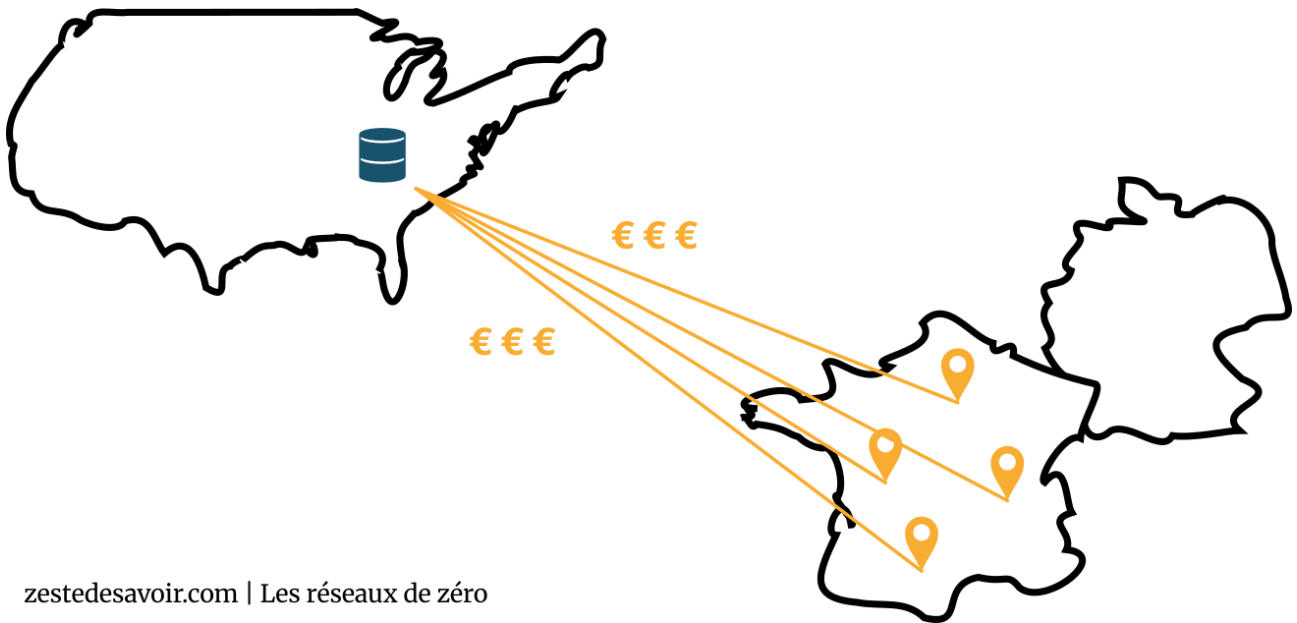


FIGURE VII.1.6. – Transmettre plusieurs fois les mêmes données des États-Unis vers la France représente un coût (CC BY)

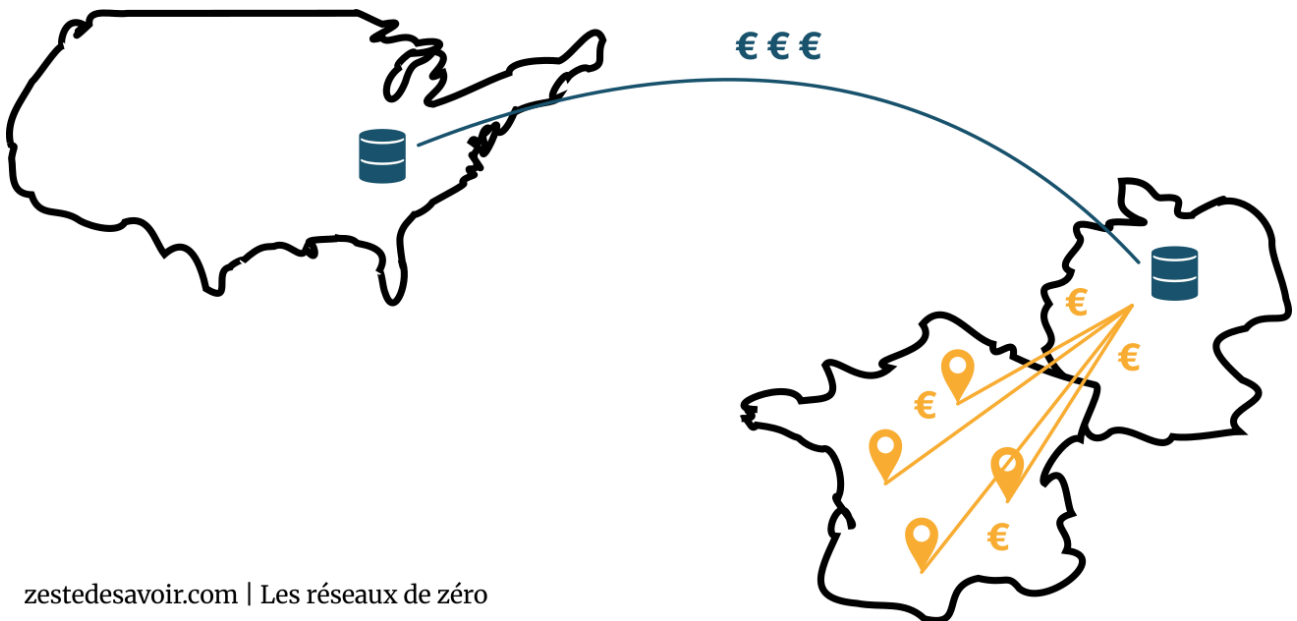


FIGURE VII.1.7. – Ce coût peut être réduit en transmettant les données une seule fois à un endroit plus proche des destinataires finaux (CC BY)

Plus récemment, la notion d'*edge computing* renvoie au fait de recourir à des équipements de traitement ou pré-traitement des données dans l'Internet des objets. Un réseau de capteurs a été déployé sur un terrain pour réaliser des analyses du sol ? Une voiture exploite ses caméras et indicateurs pour se garer toute seule ? Au lieu d'envoyer l'intégralité des données dans le cloud, dont on sait qu'une partie est inutile, on peut mettre en place un ordinateur intermédiaire dans notre réseau de capteurs pour filtrer les informations pertinentes, ou on peut déployer un système d'analyse localement dans le véhicule pour ne transmettre dans le cloud que ce qui ne peut pas être calculé localement.

1. On emploie aussi parfois les termes d'évolutivité ou d'extensibilité.

## Conclusion

Dans ce chapitre, nous n'avons fait qu'effleurer les concepts de base du cloud computing. Il y a bien plus de choses à dire et à explorer dans cet univers, suffisamment pour y dédier des livres entiers. Pour approfondir ces notions, nous vous suggérons les séries de vidéos de la chaîne [Cookie Connecté](#) sur Youtube : "Cloud : les concepts les plus importants" et "Cloud Native : les concepts les plus importants" .

## VII.2. Software Defined Networking: le réseau à la carte

### Introduction

À en croire certains ouvrages, le *Software Defined Networking* est l'avenir du réseau. Ou même le présent. Nous nous devons donc de traiter ce sujet ! De quoi s'agit-il ? Eh bien... personne n'est d'accord sur la définition. 🍌 En fait, ce terme regroupe plusieurs choses : la séparation contrôle / données, la "programmabilité" du réseau, l'utilisation de protocoles comme OpenFlow, la virtualisation de composants réseaux, l'utilisation de matériel dont les interfaces sont configurables par API, ... Cette liste est loin d'être exhaustive. Nous allons traiter plusieurs points qui, selon nous, rentrent dans le cadre du **SDN**, en répondant à des questions comme : qu'est-ce que c'est, pourquoi ils existent, à quels besoins ils répondent, et comment on les met en place.

#### VII.2.1. Un réseau plan-plan

Le principal concept du SDN qui fait consensus est **la séparation entre le plan contrôle et le plan données**. Quand on y pense, un équipement réseau quel qu'il soit opère toujours une suite d'actions définie :

1. Il reçoit des données sur une interface.
2. Il détermine ce qu'il en fait : les renvoyer sur telle autre interface, sur toutes les autres interfaces, ne pas les renvoyer du tout, modifier un port ou une adresse, changer le contenu niveau applicatif, ...
3. Il renvoie les données (modifiées ou non) sur les interfaces déterminées à l'étape précédente.

Essayons d'appliquer cette décomposition à des équipements que nous avons vus dès le début du cours. Un **hub** reçoit des données sur une interface (étape 1), "détermine" qu'il va les renvoyer telles quelles sur toutes les autres interfaces (étape 2) et procède à cette retransmission (étape 3). Dans ce cas, l'étape 2 est presque accessoire, puisqu'un hub n'a pas d'"intelligence", il ne prend pas de décision, il a été programmé pour toujours réaliser la même action. Ce n'est pas le cas du **switch Ethernet** : il reçoit des données sur une interface (étape 1), il identifie vers quelle(s) interface(s) elles doivent être renvoyées en fonction de l'adresse MAC de destination, des VLAN, et d'éléments de configuration tels que le *port mirroring*, puis les renvoie vers l'interface ou les interfaces concernée(s) le cas échéant (étape 3). Même raisonnement pour le **routeur**. L'étape 1 est toujours la même, nous n'allons pas la répéter à chaque fois. Le routeur analyse sa table de routage pour déterminer quel sera le prochain saut en fonction de l'adresse IP de destination, il opère aussi éventuellement des modifications d'adresses s'il est configuré pour faire du NAT, il vérifie si le trafic est autorisé ou non par des règles de contrôle, et tout un tas d'autres cas prévus dans sa configuration (étape 2). Après tout ce travail, il peut enfin procéder à l'envoi des

données sur la ou les interfaces déterminées (étape 3). Vous avez compris l'idée, c'est la même chose pour les pare-feux et tout autre équipement réseau.

Les étapes 1 et 3 peuvent être considérées comme faisant partie d'un plan "données", puisqu'il s'agit d'envoyer ou de recevoir des données, tandis que l'étape 2 constitue un plan "contrôle". L'approche SDN propose de décorréliser ces plans et de les voir comme des processus distincts.

?

Hein ? Pourquoi vouloir séparer des étapes d'un tout cohérent ? On a besoin de tout ça pour faire fonctionner le réseau !

Effectivement, mais a-t-on besoin que l'étape 2, qui détermine ce qui est renvoyé et vers où, soit réellement réalisée par le même équipement physique qui fait transiter les données ? Techniquement, on peut proposer une approche différente. Que diriez-vous de centraliser l'intelligence des équipements dans un serveur dédié ? Regardez ce que cela pourrait donner.

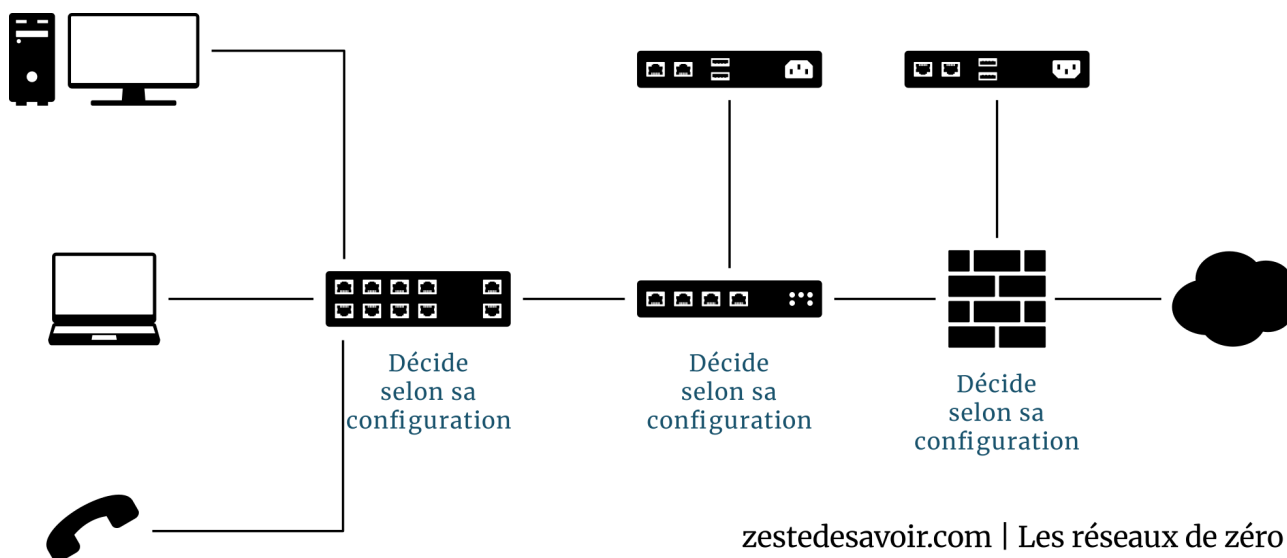


FIGURE VII.2.1. – Cas « classique » : chaque équipement agit selon sa configuration propre (CC BY)

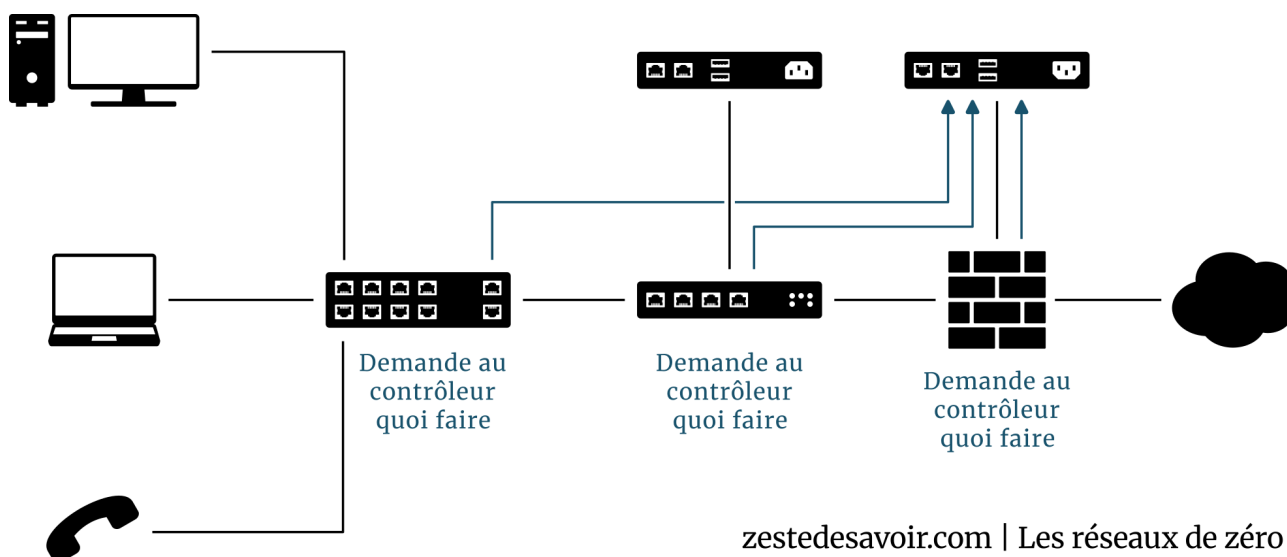


FIGURE VII.2.2. – Cas « centralisé » : un contrôleur prend tout ou partie des décisions à la place des équipements (CC BY)



Ça a vraiment un intérêt ?

Dans certains cas, oui. Il existe différentes gammes de routeurs, de switches, etc. chez chaque constructeur d'équipements, qui sont adaptés à des environnements techniques différents : petite entreprise, datacentre, réseaux WAN, etc. Il peut y avoir des différences physiques, comme le nombre de ports, le type d'interface (fibre optique, Ethernet, série, ...), mais une partie de ces différences est logicielle et impacte la façon dont l'équipement est administré, ses performances pour réaliser certaines opérations comme du chiffrement ou encore les fonctionnalités qu'il propose. Centraliser les opérations de contrôle sur un serveur permet de ne pas se soucier de tout cela, puisque tout est géré par une machine unique. Déporter la prise de décision vers un contrôleur n'est possible que sur des équipements conçus pour cela, dont le coût peut être plus élevé que pour des machines plus classiques. La mise en place de ce type de SDN n'est pas forcément économique et répond à des besoins précis, notamment d'évolutivité. Nous avons évoqué ce terme dans le chapitre sur le cloud, et ce n'est pas un hasard : c'est ce milieu qui est le plus susceptible d'avoir recours à de telles techniques, étant donné que les besoins des clients peuvent rapidement changer.



Ce que nous décrivons ici est un concept général, ses implémentations peuvent varier. Par exemple, il est possible pour le contrôleur de donner des instructions de prise de décision aux équipements pour des cas précis, comme le ferait un administrateur réseau en configurant le matériel, tout en faisant remonter les cas non définis pour laisser le contrôleur prendre la décision.

### VII.2.1.1. Gestionnaire de trames

Un autre intérêt est celui de faciliter le management. Nous avons évoqué deux plans : celui des données et celui du contrôle. Dans le milieu du SDN, on parle aussi parfois d'un troisième plan : le **management**. Si le plan contrôle se retrouve sur un serveur, on n'a plus à réaliser de configuration à grands coups de ligne de commande sur les routeurs et autres switches, à part pour leur dire "référez-vous à tel serveur". On peut ainsi programmer le serveur en lui donnant des instructions pour aboutir à un résultat tel que nous le percevons, plutôt que de devoir apprendre la façon de s'adresser à nos équipements. Les images suivantes illustrent la différence d'approche.<sup>2</sup>

General properties

☐

 Flow name

ADDED ☒

 Table

ADDED ☒

 ID

ADDED ☒

 Hard timeout

ADDED ☒

 Idle timeout

ADDED ☒

 Cookie

☐

 Cookie mask

ADDED ☒

 Priority

Match

ADDED ☒

 In port

☐

 Metadata

☐

 Metadata mask

☐

 Ethernet type

☐

 Source MAC

☐

 Destination MAC

☐

 Vlan ID

☐

 Vlan priority

+

Config ☐

Operational ☒

Device openflow:262548554233160 [None] [Open vSwitch]

General properties

Table 0

ID #UF\$TABLE\*0-49

Priority 2

Hard timeout  X

Idle timeout  X

Cookie  X

In port  X

Actions

Output port X

FIGURE VII.2.3. – Capture d’écran de l’interface d’OpenDayLight, un outil pour gérer graphiquement le réseau

```

hostname ROUTEUR-CISCO-2911
enable secret 5 $1$mERr$5.a6P4JqbNiMX01usIfka/
ip dhcp excluded-address 192.168.10.1 192.168.10.10
ip dhcp excluded-address 192.168.20.1 192.168.20.10
ip dhcp pool LAN-POOL
  network 192.168.10.0 255.255.255.0
  default-router 192.168.10.1
  dns-server 8.8.8.8
ip dhcp pool GUEST-POOL
  network 192.168.20.0 255.255.255.0
  default-router 192.168.20.1
  dns-server 8.8.4.4
ip cef
no ipv6 cef
username admin secret 5 $1$mERr$AFX/pZT1Lh7NP3Dp3P/qq/
license udi pid CISCO2911/K9 sn FTX1524VC29-
ip ssh version 2
ip domain-name exemple.local
spanning-tree mode pvst
interface GigabitEthernet0/0
  description Connexion au WAN
  ip address 192.168.1.1 255.255.255.0
  duplex auto
  speed auto
interface GigabitEthernet0/1
  description Connexion au LAN
  ip address 192.168.10.1 255.255.255.0
  duplex auto
  speed auto
interface GigabitEthernet0/2
  description Reseau Invits
  ip address 192.168.20.1 255.255.255.0
  duplex auto
  speed auto
interface Vlan1
  no ip address
  shutdown
ip classless
ip route 0.0.0.0 0.0.0.0 192.168.1.254
ip flow-export version 9
access-list 100 permit ip any any
line con 0

```

FIGURE VII.2.4. – Extrait d'une configuration en ligne de commande d'un routeur Cisco

Prenons un exemple concret : nous souhaitons interdire tout trafic HTTP et HTTPS en provenance du réseau 10.0.0.0/8 dans le cadre de l'infrastructure du schéma ci-dessous. L'approche classique consiste à commencer par déterminer où est ce réseau, sur quel équipement on implémente ce besoin, quelles sont les lignes de commande à saisir (cela n'a rien à voir de configurer du Cisco ou du Fortinet ou du Juniper, il y a même des différences entre les modèles d'un même constructeur), et au moindre changement, tout est à refaire. L'approche SDN consiste à saisir sur le contrôleur, par exemple au moyen d'une interface web, "interdire tout trafic HTTP et HTTPS en provenance de 10.0.0.0/8". C'est le serveur qui détermine comment cela se traduit réellement

## VII. Évolutions

pour les équipements. Avec ce cas de figure, il est même possible que le plan contrôle ne soit pas centralisé mais uniquement le plan management : le serveur envoie alors des instructions aux équipements concernés au travers d'un protocole défini en amont (HTTP avec une API REST, OpenFlow, ...).

Là encore, cette approche peut répondre à certains besoins, mais elle n'est pas applicable partout et n'est pas une solution miracle.

### VII.2.1.2. Le protocole OpenFlow

OpenFlow se veut être le standard<sup>1</sup> pour la séparation des plans données et contrôle. Il a été créé à la fin des années 2000 et est en constante évolution. Il permet de transmettre aux équipements réseau des instructions de traitement en fonction du paramétrage du contrôleur SDN, mais aussi de transférer au contrôleur des paquets ou des trames dans le cas où l'équipement ne dispose pas d'informations suffisantes pour prendre une décision.

### VII.2.2. Tout un programme

On inclut parfois dans le SDN la virtualisation d'équipements réseau. Nous allons évoquer ce sujet, ainsi que celui de la virtualisation de fonctions réseau.



Des équipements réseau virtuels ? Et comment on réalise les raccordements ?

Eh bien, avec des câbles virtuels ! 🍌 En fait, on trouve ce cas de figure dans certaines infrastructures, comme les **hyperviseurs**. Ce terme désigne un programme qui orchestre plusieurs machines virtuelles et leur permet de communiquer entre elles et avec la machine hôte. VMware Workstation, VirtualBox ou encore Proxmox sont des hyperviseurs. Pour fournir des services réseau (routage, NAT, commutation ...), on peut faire appel à une machine virtuelle qui va faire office de pare-feu, par exemple. Elle sera en charge de router les paquets, filtrer les flux, établir des liens VPN, etc. **PfSense** est un système d'exploitation répandu sur les infrastructures virtualisées pour servir de pare-feu. On peut aussi imaginer déployer un switch virtualisé, avec un système comme **Open vSwitch**<sup>1</sup>.

L'utilisation de la virtualisation de fonctions réseau (NFV, Network Functions Virtualization) trouve son intérêt dans le déploiement d'infrastructures cloud lourdes. On peut avoir besoin de VLAN, de répartition de charge, de filtrage, etc. même dans des architectures virtualisées. Les fonctionnalités embarquées dans les hyperviseurs peuvent se révéler limitées face à des besoins conséquents.

## Conclusion

De nos jours, tout système informatique peut être virtualisé. Plus besoin d'infrastructures lourdes et coûteuses... à part pour les fournisseurs de services cloud. Et encore, avec certains

---

1. Ce standard propose une spécification ouverte pour que l'implémentation soit possible. Elle est disponible [au format PDF](#) 📄.

2. Ces captures sont purement illustratives. Celle d'OpenDayLight est issue de la vidéo "[GNS3 Talks : Docker, Open vSwitch, SDN and OpenFlow Part 3 : GNS3 Switching Part 9](#) 📄" de David Bombal sur Youtube.

1. Pour en savoir plus, la [page officielle](#) 📄 est assez complète bien que très technique.

## *VII. Évolutions*

outils tels que vSphere, il est carrément possible de virtualiser des datacentres entiers ! La seule limite est la puissance de l'infrastructure physique, qui doit pouvoir supporter les services fournis. Le SDN est potentiellement présent partout, de manière invisible, mais il ne remplace pas les systèmes existants, il permet surtout de les virtualiser. Son utilisation devient un pan à part entière du monde du réseau, qui s'ajoute aux métiers plus traditionnels d'administration et d'ingénierie réseau qui, maintenant, peuvent s'opérer à distance sur des machines virtuelles.

# Conclusion

Quelles sont les prochaines évolutions à venir ? Difficile à dire tant le milieu des réseaux évolue rapidement. Nous essaierons de maintenir cette partie à jour en fonction du développement et du déploiement de nouveautés.

## Huitième partie

### Quelques notions de sécurité réseau

# Introduction

Cette partie contient quelques notions de sécurité réseau, comme son nom l'indique. Les chapitres qui y sont publiés sont indépendants du reste du cours. Ils ne nécessitent pas de connaissances approfondies en réseaux et peuvent être feuilletés à n'importe quel moment.

## VIII.1. Introduction à la sécurité

### Introduction

Commençons cette partie par une introduction aux principes de sécurité. Nous allons d'abord voir ce qu'est la sécurité et les différentes façons de sécuriser un réseau. Ce chapitre servira de base aux autres. Dans ceux-ci, nous entrerons un peu plus dans les détails des principales attaques en réseau et nous verrons comment les prévenir. 🍌

i

Cette partie n'aborde que quelques notions de cybersécurité. Pour aller **beaucoup** plus loin, vous pouvez vous procurer **La Cybersécurité de Zéro** [↗](#), disponible sur toutes les plateformes et dans les bonnes librairies !



FIGURE VIII.1.1. – La Cybersécurité de Zéro, aux éditions Eyrolles, reprend des éléments de cette partie

### VIII.1.1. C'est quoi la sécurité ?

Bien ! Nous allons consacrer cette partie à la sécurité réseau... OK... Mais, à quoi sert la sécurité ? Le savez-vous ?

Un réseau est constitué de plusieurs nœuds. D'ailleurs, c'est même cela la définition basique d'un réseau : une interconnexion de nœuds. Mais, ces nœuds sont interconnectés dans le but de s'échanger des informations, qui sont appelées des **ressources**.

?

Qu'est-ce qu'une ressource ?

Dans un réseau, une ressource est toute forme de données ou d'applications que l'on peut utiliser pour accomplir une tâche précise. Par exemple, dans le cas d'une imprimante, les ressources dont elle a besoin pour accomplir sa fonction (imprimer) sont majoritairement le papier et l'encre. Dans une entreprise, le réseau a principalement pour but l'échange de ces ressources desquelles

dépendent les activités commerciales de l'entreprise. Étant donné que ces ressources sont cruciales à son bon fonctionnement, il est important, voire obligatoire, de veiller à leur sécurité.



Pourquoi protéger les ressources ?

En ce qui concerne l'imprimante, nous savons tous que son travail consiste à imprimer des données numériques sur support physique. Que se passerait-il alors si quelqu'un volait tous les papiers de l'imprimante ? Elle a beau avoir l'encre, sans papier elle ne peut pas remplir sa fonction. On peut ainsi parler du **niveau de dépendance** ou **degré de dépendance** de l'imprimante. Non seulement elle a besoin d'encre, mais aussi de papier. Il est donc important de veiller à ce que personne n'enlève l'encre ou le papier de l'imprimante, car ces deux ressources lui sont liées. 🍏

C'est logique vous nous direz, mais nous sommes en train de poser des bases très importantes en réseaux. Vous **devez** saisir cette notion de dépendance. Vous aurez beau protéger une ressource importante, si une autre ressource qui est liée à cette dernière est exposée, cela demeure une faille qu'un intrus peut exploiter. La sécurité consiste à veiller à l'intégrité des ressources d'un réseau, ainsi qu'à leur disponibilité.



Cette information est très importante, c'est une règle que vous devez retenir. Plus des nœuds d'un réseau dépendent d'une ressource, plus cette ressource sera la cible principale d'une attaque. Plus le degré de dépendance est élevé, plus il faut songer à sécuriser ladite ressource.

### VIII.1.1.1. Étude de cas : Zeste de Savoir

Imaginez que vous administrez le réseau de Zeste de Savoir. Commençons d'abord par localiser les ressources du site.



Quelles sont les ressources de Zeste de Savoir ?

Elles sont nombreuses, mais voici les principales :

- Les tutoriels ;
- Les articles.

Les membres du site peuvent être considérés comme étant des ressources, mais nous allons plutôt les considérer comme étant les nœuds qui forment le réseau de Zeste de Savoir. Les tutoriels sont, sans l'ombre d'un doute, les ressources les plus importantes. Ce sont ces ressources qui génèrent majoritairement tout le trafic du site.



Et si tous les tutoriels étaient supprimés ?

Que se passerait-il si un jour vous vous connectiez sur Zeste de Savoir et qu'il n'y ait plus un seul tutoriel en ligne ? 🍏

Zeste de Savoir gardera encore longtemps ses membres, mais le trafic va considérablement diminuer ou stagner. Grâce à la communauté, il y aura toujours des membres qui viendront

## VIII. Quelques notions de sécurité réseau

pour discuter sur les forums. Mais, la principale ressource attrayante n'étant plus disponible, Zeste de Savoir commencera à entrer dans une période de déclin.

Une fois qu'un « attaquant » (*hacker* si vous voulez) localise la ressource dont le réseau dépend considérablement, cette dernière deviendra sa cible principale (on parle aussi de cible d'évaluation). C'est alors que nous répétons notre règle d'or :



Plus des nœuds d'un réseau dépendent d'une ressource, plus cette ressource sera la cible principale d'une attaque. Plus le degré de dépendance est élevé, plus il faut songer à sécuriser ladite ressource.



Vous devez noter que « nuire » aux tutoriels implique l'exploitation d'une faille quelconque. Nous allons parler de l'exploitation dans les sous-parties suivantes.

Tout au long de cette partie, nous allons vous apprendre les bonnes pratiques à mettre en place pour maximiser la sécurité des ressources dans un réseau.

Si vous avez lu tout le cours, vous êtes très riches en connaissances. Nous allons donc entrer dans des détails techniques (surtout dans le langage). Nous allons vous considérer comme étant des professionnels travaillant en tant qu'administrateurs réseaux. Ainsi, vous aurez à votre disposition de nombreux conseils en sécurité, et surtout beaucoup de techniques de prévention et de protection contre les attaques. 🍊

### VIII.1.2. Comprendre la terminologie

Nous allons, dans cette sous-partie, aborder la terminologie en sécurité. Oui, ça ne paraît pas extraordinaire, mais ce premier chapitre n'est qu'une introduction au concept. 🍊

#### VIII.1.2.1. Une menace

En anglais *threat*, une menace est une situation qui pourrait potentiellement conduire à un événement dramatique. Quand vous recevez des appels anonymes d'une personne menaçant de vous tuer, ceux-ci constituent une situation qui pourrait conduire à un événement dangereux, votre mort en l'occurrence. Dans la vie courante, lorsque vous recevez des menaces, vous prenez des mesures en conséquence, par exemple informer la police.

Dans un réseau, le principe reste le même. Une menace est une situation qui pourrait conduire à une potentielle rupture de la sécurité de votre réseau. Dès que vous localisez une menace, si innocente paraisse-t-elle, si vous êtes un employé, informez immédiatement votre administrateur de sécurité. Si c'est vous l'administrateur, commencez à mettre en place toute solution pouvant neutraliser cette menace.

#### VIII.1.2.2. La vulnérabilité

Également appelée **faille**, la vulnérabilité est une faiblesse quelconque dans votre réseau qu'un *hacker* peut exploiter au travers d'un logiciel qu'on appelle « exploitateur ». Ces logiciels sont des morceaux de code qui profitent de la présence de *bugs* dans un système ou de failles dans un réseau pour avoir accès illégalement à des ressources, voire augmenter les privilèges d'un compte utilisateur afin de mieux exploiter les ressources et ouvrir une porte pour un autre *hacker*.

La majorité des failles sont des choses que l'on néglige, telle que la complexité d'un mot de passe. En tant qu'administrateur en sécurité réseau, il est important d'informer régulièrement les employés de votre société sur les politiques de sécurité, c'est-à-dire les bonnes pratiques. Parmi ces bonnes pratiques figurent la complexification d'un mot de passe. Si un employé junior0 a pour compte utilisateur « junior0 » et mot de passe « junior0 », cette faiblesse est une faille qu'un *hacker* peut exploiter pour avoir accès aux ressources sous le pseudo de junior0. Une fois qu'il a accès à ce compte, il peut éventuellement augmenter ses privilèges, par exemple passer d'un compte utilisateur simple à un compte administrateur et avoir les mêmes privilèges que vous, l'administrateur légitime. 🍊

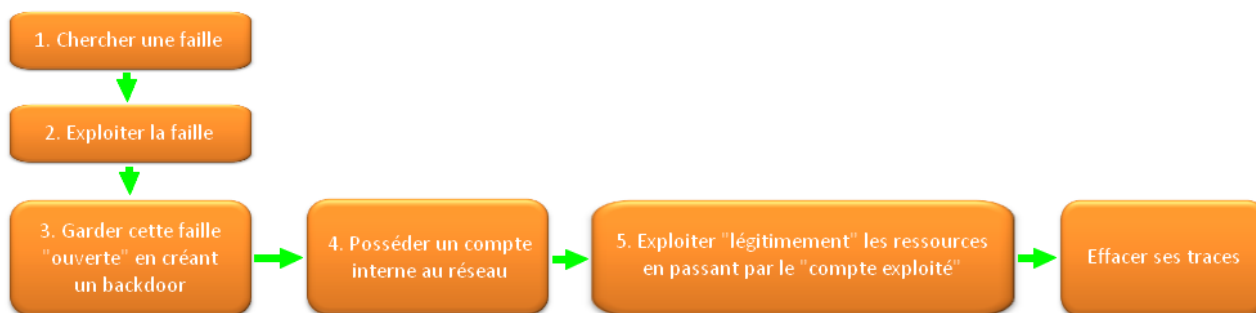
Il existe principalement deux types d'exploitation :

- **locale** : une exploitation est dite locale lorsqu'elle émane de l'intérieur du réseau. Comme nous l'avons dit plus haut, si un *hacker* se sert du compte utilisateur d'un nœud légitime de votre réseau pour augmenter ses privilèges, toute attaque à partir de ce compte sera locale.
- **distante** : une attaque distante, contrairement à une attaque locale, ne vient pas du réseau en tant que tel. C'est un *hacker* qui se trouve dans un réseau distant qui essaie d'exploiter une faille « à distance » sans avoir accès à un compte local à votre réseau.

Que cela vous surprenne ou non, la majorité des attaques sont locales et non distantes.

### VIII.1.2.3. Les étapes d'exploitation d'une faille

Voici un schéma illustrant une procédure très commune d'exploitation d'une faille :



Les réseaux de zéro - zestedesavoir.com

FIGURE VIII.1.2. – Schéma d'exploitation d'une faille

Le schéma est très évocateur, mais nous allons quand même le survoler un peu.

- **L'étape 1** consiste à trouver une faille. Généralement on utilise des logiciels de scannage (de port par exemple). Il est important de commencer par trouver la cible d'évaluation. Il faut une stratégie pour remporter une guerre. La cible d'évaluation est expliquée plus bas.
- **L'étape 2** consiste, bien sûr, à exploiter cette faille, c'est-à-dire, profiter de cette ouverture pour aboutir à « quelque chose ».
- **À l'étape 3**, on s'assure que cette faille restera toujours ouverte afin de pouvoir l'utiliser à chaque fois. Eh oui, c'est comme l'*open source*, c'est mieux de réutiliser un code que de tout coder soi-même. 🍊 Si une faille est déjà existante, autant la maintenir plutôt que d'en chercher une autre. Cela se fait souvent par l'installation d'une porte dérobée (*backdoor*).

- **L'étape 4**, c'est « faire quelque chose » de cette faille. Par exemple, « prendre possession d'un compte » (*to take ownership* en anglais) et augmenter les privilèges de ce compte. Ainsi, si c'était un compte utilisateur simple, il s'agit d'acquérir les privilèges d'un compte administrateur.
- **L'étape 5** consiste à exploiter « localement » les ressources. Étant donné que vous possédez un compte interne au réseau, vous passez pour une personne légitime. C'est le principe de l'exploitation interne ou locale.
- **Finalement**, la plus subtile des étapes, c'est, bien sûr, effacer ses traces. En termes d'investigation, c'est ce qu'on appelle « commettre le crime parfait ». Nous verrons comment effacer les traces dans un réseau et aussi quelques techniques d'évasion (fuite) en cas de détection.



Nous avons volontairement « disséqué » les étapes d'exploitation, mais sachez que ces 6 étapes peuvent être classées en 3 catégories. L'étape 1 c'est ce qu'on appelle la phase de « test de pénétration » ou « papier-crayon ». C'est une phase dans laquelle on se documente au maximum et on planifie notre attaque en cherchant à découvrir les caractéristiques (topologie, plan d'adressage, etc) du réseau que l'on cible afin de trouver une faille. Les étapes 2 à 5 peuvent être groupées dans une catégorie « exploitation ». En effet l'exploitation d'une faille couvre tous les aspects mentionnés dans ces étapes. La dernière étape peut être classée dans la catégorie « évasion ». « Effacer ses traces », c'est une forme de « fuite subtile ».

### VIII.1.2.4. Cible d'évaluation

Le terme **cible d'évaluation** vient de l'expression anglaise *Target Of Evaluation (TOE)*. Cette expression est plus propre à la certification CEH (que nous allons vous présenter) qu'à la sécurité de façon générale. Mais, le principe est universel. Une cible d'évaluation est la ressource la plus importante, celle dont dépendent les nœuds de votre réseau. Pour Zeste de Savoir, nous avons vu qu'il s'agissait des tutoriels. Ils sont, en termes de sécurité, une cible d'évaluation pour le *hacker*. Pour les banques, la cible d'évaluation pourrait être, par exemple, les numéros de comptes des clients.

### VIII.1.2.5. Qu'est-ce qu'une attaque ?

On appelle attaque, toute tentative maligne de pénétrer un réseau illégalement, par le biais de l'exploitation d'une vulnérabilité (ou faille). La majorité des attaques en réseau sont dues aux failles, et pour cela, le coupable numéro 1 c'est vous-même.

Imaginez que vous avez sur votre compte bancaire 300 000 euros. Pour ne pas oublier votre mot de passe, vous avez écrit sur un petit morceau de papier « mot de passe de mon compte : 4590 ». Un jour vous oubliez ce morceau de papier sur votre bureau au travail avec votre portefeuille. Un collègue s'en rend compte, prend votre carte, va dans un guichet retirer une grosse somme et retourne au bureau déposer votre portefeuille. Le fait d'avoir écrit ce mot de passe était une faille de sécurité. Avoir oublié le portefeuille était une erreur d'attention. Le retrait illégal de l'argent était une attaque. Qui est le premier coupable ? Celui qui a volé l'argent ou vous-même ?

En sécurité, la plupart des attaques sont dues à ce couple « faille-erreur ». Une faille est une ouverture et l'attaquant profite de votre erreur pour exploiter la faille.

### VIII.1.2.6. Identifier / Authentifier / Autoriser

L'autorisation est différente de l'identification. C'est en quelque sorte une alternative. Lorsque vous avez un système qui ne nécessite pas la divulgation de l'identité de la personne, vous n'avez besoin que de vérifier si la personne remplit les critères lui donnant l'autorisation d'exécuter une opération, peu importe qui elle est. Par exemple, lorsque vous voulez retirer de l'argent d'un guichet électronique, le guichet ne vous identifie pas au sens strict du terme, mais il vous autorise. L'autorisation est une forme d'identification. La différence est fine, mais cet exemple vous aidera à mieux comprendre.

M. Pierre détient une carte qui a un numéro : 1428 6983 0596 et un code 0978. Il décide d'aller retirer 100 euros dans un guichet électronique. Lorsqu'il insère la carte et tape le code de sécurité, le guichet « sait » que la transaction est demandée par M. Pierre, parce que c'est sa carte. On peut l'identifier par le numéro de la carte. Le code de sécurité permet de l'authentifier premièrement, ensuite de l'autoriser à faire une transaction.



L'autorisation doit toujours être précédée de l'authentification.

En résumé, lorsque M. Pierre insère la carte, le système du guichet identifie par le numéro de la carte que c'est M. Pierre qui essaie de faire une transaction. Le numéro de la carte permet de l'identifier. Mais qu'est ce qui nous prouve que c'est bien M. Pierre ? Il faut lui demander le code de sécurité, car c'est personnel, et lui seul doit le détenir. Ainsi nous lui disons « Prouve-moi que tu es bien M. Pierre. Prouve-moi que tu es celui que tu dis être en me donnant le code de sécurité ». Pierre entre alors le code 0978, le guichet l'authentifie, atteste que la personne essayant de faire la transaction est authentique, c'est bel et bien M. Pierre. Par la suite, il peut effectuer les transactions de son choix, car il a désormais l'**autorisation**.

Maintenant un jour, il envoie son fils Matthieu. Ce dernier va retirer de l'argent étant en possession de la carte de son père et du code de sécurité. Le guichet lui donnera également accès, lui autorisera à faire la transaction, sans nécessairement identifier qu'ici, il s'agit de Matthieu et non de Pierre. Voyez-vous ?

Si ce n'est pas toujours clair, soyez tranquille, nous avons plusieurs autres exemples. Commençons d'abord par définir ces termes un à un avec un exemple à l'appui.

#### VIII.1.2.6.1. Identifier

C'est l'action d'évaluer l'identité d'une personne. Quand ce verbe est pronominal (s'identifier), c'est l'action de décliner son identité. Par exemple, si vous allez un jour au bureau d'une société, appelons-la ZesteCorp, rencontrer la secrétaire, vous allez vous présenter : « Bonjour, je m'appelle Pierre ». Vous vous identifiez. Si je veux rencontrer une personne que je n'ai jamais vue, je peux lui expliquer comment m'identifier, par mon habillement, par exemple : « Je serai vêtu d'un jean noir et d'une chemise blanche avec un chapeau de cowboy ». L'ensemble de ces descriptions servira à mon identification.

#### VIII.1.2.6.2. Authentifier

C'est l'exigence d'une preuve irréfutable qu'une personne est bel et bien celle qu'elle dit être. Pour reprendre l'exemple d'une visite à ZesteCorp, si la secrétaire vous demandait de prouver que vous êtes bel et bien Pierre, vous montrerez votre carte d'identité par exemple. La carte constitue donc une évidence, une preuve que vous êtes celui que vous prétendez être.

## VIII. Quelques notions de sécurité réseau

Dans une rencontre « face à face », l'identification et l'authentification se font très souvent au même moment, cependant dans des systèmes « distants » (où nous n'avons pas d'interaction face à face), ces deux étapes sont séparées.

### VIII.1.2.6.3. Autoriser

L'autorisation consiste à vérifier qu'une personne identifiée a le droit de faire telle ou telle chose. Par exemple quand vous êtes majeur, vous avez le droit de voter, l'autorisation de voter. L'autorisation est la dernière étape dans un système de contrôle d'accès. Vous vous identifiez premièrement, vous donnez une preuve permettant de vous authentifier, et lorsque le système évalue cela, il vous donne l'accès, l'autorisation.

Si Pierre veut se connecter dans le réseau local de l'entreprise dans laquelle il travaille, il donnera son nom d'utilisateur (pour son identification), il donnera son mot de passe (pour son authentification), mais il faudra que le système détermine ses droits d'accès.

À votre avis, pourquoi lorsque vous vous connectez sur Zeste de Savoir, vous n'avez pas accès à l'interface de modération par exemple ? Parce que le système de sécurité vérifie vos privilèges dans le domaine, et vous autorise en fonction des droits que vous avez. Si vous êtes administrateur, vous avez le droit d'un membre, d'un modérateur, d'un validateur et d'un admin. Si vous n'êtes qu'un membre, vos actions (transactions) sur le site seront limitées en vertu de vos droits. C'est à ça que sert l'étape d'autorisation, limiter les possibilités de vos actions selon les privilèges que vous détenez.

Voilà, de manière simple, comment comprendre ces 3 notions fondamentales.

## VIII.1.3. Les moyens de sécurité

Nous savons maintenant ce qu'est une faille, qu'est-ce qu'une attaque, une exploitation (locale ou distante), etc. Ok, tout cela c'est bien, mais maintenant quels sont les moyens à notre disposition pour sécuriser notre réseau ? Il existe deux façons de sécuriser un réseau. Ces deux façons sont liées l'une à l'autre et une bonne politique ou implémentation de sécurité résulterait d'un mélange de ces deux moyens de protection.

### VIII.1.3.1. La sécurité physique

La sécurité physique est obligatoire. C'est d'ailleurs la première des choses à laquelle vous devez penser avant tout ce qui est *high-tech*. Parfois on se laisse tellement emporter par le prestige de la technologie qu'on oublie la base des bases. À quoi ça vous servira d'avoir sur votre serveur des logiciels hyper sécurisés si n'importe qui peut y accéder et débrancher les câbles ? 🍊 La règle à retenir est la suivante :

**La sécurité technologique dépend de la sécurité physique.**

En fonction de vos besoins et de vos moyens, les solutions sont nombreuses. Cela peut aller d'un simple local fermé à clé, à un bâtiment vidéo-surveillé protégé par des barbelés, en passant par des gardiens et des portes blindées... Déjà, des machines sensibles sous clé, c'est pas mal. 🍊

### VIII.1.3.2. Les techniques de contrôle d'accès

Là où on a souvent davantage de latitude, c'est sur l'aspect technologique de la protection. Aujourd'hui, pour s'authentifier et obtenir l'accès à des services, on a recours aux mots de passe. Cette technique pose plusieurs soucis.


## VIII. Quelques notions de sécurité réseau

D'abord, il est facile pour un programme informatique de casser un mot de passe court ou simple. À l'aide de dictionnaires ou même par force brute, un code tel que « citron42 » ou « mzdudv » ne résiste pas longtemps. Il est nécessaire de complexifier cela. Au lieu d'un mot, on peut créer une phrase de passe (avec quelques caractères spéciaux).

On ne peut toutefois pas garantir qu'un code, quel qu'il soit, ne soit pas un jour compromis. C'est pourquoi il ne faut pas utiliser le même mot de passe pour différents services. Sinon, un attaquant qui récupère vos identifiants peut accéder à tous vos comptes ! C'est là qu'est le deuxième souci. C'est pénible. On a tous plein de comptes différents pour plein de choses et retenir tous ces mots de passe relève du défi.

i

Fin 2018, une vague de *spams* a ciblé des entreprises et des internautes. Il s'agissait d'envois d'e-mails contenant, en objet, un supposé mot de passe du destinataire. Ces codes ont été retrouvés dans des bases de données piratées. Le message prétendait, en exposant ainsi un mot de passe qui aurait été utilisé par le destinataire, que des données compromettantes avaient été dérobées. Il demandait une rançon en l'échange du silence de l'attaquant.

Une solution consiste à [utiliser un gestionnaire de mots de passe](#) . Cet outil peut générer de longs mots de passe et les stocker de manière sécurisée. Pour les débloquent et les utiliser, on peut passer par les empreintes digitales ou une clé physique.

Une autre possibilité complémentaire est l'authentification par multiples facteurs. Cela consiste à fournir plusieurs preuves distinctes pour s'authentifier : un code unique envoyé par SMS, que seul le vrai propriétaire du compte doit pouvoir consulter, une carte à puce, etc. Cette technique est notamment employée par les banques pour empêcher l'utilisation frauduleuse de cartes bancaires.

## Conclusion

Maintenant, vous savez en quoi consiste la sécurité. 🍊

## VIII.2. Malins, les logiciels !

### Introduction

Dans ce chapitre, nous allons voir les différents types de logiciels utilisés pour compromettre la sécurité ou les données des systèmes informatiques. Il s'agit uniquement de vocabulaire et de terminologie. Beaucoup de personnes confondent virus, chevaux de Troie, espions et autres termes. Nous allons clarifier tout cela, ça vous sera très utile de pouvoir désigner précisément quelque chose dans une discussion.

Le terme générique qui désigne un de ces programmes est *malware*, ou en français logiciel malicieux ou encore logiciel malveillant. Nous allons voir les principaux types de *malware*.

### VIII.2.1. Des fins malveillantes

En fonction de leur objectif, les *malwares* peuvent être classés par catégories. Ainsi, un logiciel d'espionnage portera un nom différent d'un logiciel publicitaire, etc.

#### VIII.2.1.1. Les adwares : du spam directement sur votre machine !

Un *adware*, ou en français **publiciel**, est un logiciel qui affiche de la publicité sur votre ordinateur. Il n'est pas forcément illégitime : certains logiciels « gratuits » installent un *adware* pour compenser la gratuité. Mais certains le sont et affichent des publicités dans le but d'ennuyer l'utilisateur et surtout de gagner de l'argent auprès des annonceurs ! Il existe des logiciels permettant de détecter et supprimer ces *adwares*, comme Ad-Aware (qui ne fait pas que ça, d'ailleurs).

#### VIII.2.1.2. Les keyloggers : dis-moi ce que tu tapes, je te dirai que tu es ma victime

Un *keylogger*, ou **enregistreur de frappes**, est un logiciel qui enregistre ce que l'utilisateur tape au clavier et le sauvegarde dans un fichier, ou l'envoie par Internet sur un serveur. Ces logiciels sont généralement discrets et ont pour objectif d'intercepter des mots de passe, des numéros de carte de crédit, et d'autres données confidentielles.

Ces logiciels ne sont pas évidents à déceler, car ils n'ont besoin pour fonctionner que de fonctions plutôt banales, qu'un logiciel légitime peut utiliser aussi. Les logiciels anti-espions comme Spybot peuvent les déceler.

#### VIII.2.1.3. Les backdoors : c'est donc ça, ce courant d'air...

Une *backdoor*, ou **porte dérobée**, est un programme qui ouvre un accès sur votre ordinateur. Si une porte dérobée est ouverte sur votre ordinateur, on peut s'y connecter via le réseau et effectuer différentes actions définies par le développeur du programme.

Pour s'en protéger, on peut utiliser un **pare-feu** comme Netfilter sous Linux, ou Zone Alarm sous Windows. Attention, une porte dérobée fonctionnant en serveur

sera vite repérée car elle demandera d'ouvrir un port sur la machine, tandis qu'une *backdoor* fonctionnant en client demandera simplement à établir une connexion avec un serveur comme le ferait n'importe quel logiciel.

### VIII.2.1.4. Les espions : la curiosité est un vilain défaut, mais qui peut rapporter gros

Les **spywares**, ou **logiciels espions**, sont des programmes qui espionnent la machine. Ils peuvent espionner votre historique de navigation, vos fichiers personnels, dérober vos mots de passe, etc.

On peut les contrer avec des anti-spywares (anti-espions) comme Spybot.

### VIII.2.1.5. Les trojans : ils s'invitent tous seuls, c'est trojantil !

Ou pas. 🍊 Un **trojan**, ou **cheval de Troie**, est un logiciel en apparence légitime (utilitaire, jeu, ...) mais qui inclut une fonction d'infiltration qui permet d'installer des espions ou autres en tous genres sur l'ordinateur.

Les *trojans* peuvent être détectés par les logiciels antivirus comme Antivir, BitDefender, Kaspersky, ...

À propos d'antivirus, tous ces programmes malveillants ne sont pas tous des virus. En effet, un virus se réplique. Mais ce n'est pas le seul, voyons cela dans la sous-partie suivante !

## VIII.2.2. Ils n'ont pas volé leurs répliques : les virus et les vers

Voyons des programmes qui se répliquent : les **virus** et les **vers**. Ils sont caractérisés par leurs façons de se dupliquer et ne sont pas par définition malveillants. Voyons leurs différences !

### VIII.2.2.1. Le ver : il se duplique pour ne pas être solitaire

Un ver est un programme qui se duplique tout seul. Quand il s'exécute, il va se copier aux endroits choisis par la personne qui l'a programmé. Il peut se dupliquer par le réseau, par exemple grâce au partage de documents, ou encore en s'envoyant par e-mail aux adresses qu'il aura collectées (le célèbre ver « [I love you](#) » en est un exemple). La plupart des vers servent à des fins malveillantes (espionnage, ...). Les logiciels antivirus sont capables de les détecter s'ils ne sont pas assez discrets. Par exemple, si un ver tente de remplacer un programme système, comme le programme « shutdown » qui commande l'extinction de l'ordinateur (sous Windows et Linux, du moins), il aura tôt fait de se faire repérer ! 🍊

### VIII.2.2.2. Le virus : il se cache pour ne pas se faire virer

Dans la vie réelle, un virus est une maladie qui profite du corps dans lequel il se trouve pour se transmettre à d'autres. En informatique, c'est un peu pareil.



Beaucoup de personnes emploient le mot « virus » pour désigner un logiciel malveillant, ce qui est incorrect. Le terme à utiliser de manière générique est **malware**.

Un virus, contrairement à un ver, n'est pas un fichier exécutable mais un **morceau de code** destiné à être exécuté et à se répliquer.

Le virus va **infecter** un fichier qui peut être de n'importe quelle nature (image, son, ...) en se copiant dans ce fichier. Il faut que le programme avec lequel le fichier infecté va être ouvert possède une faille pour que le virus puisse s'exécuter et se répliquer dans d'autres fichiers. Ça vous paraît abstrait ? Alors voici un exemple.

Paul reçoit par e-mail une photo au format jpg. Jusqu'ici, rien d'anormal. Sauf que Paul n'a pas mis à jour son système depuis un certain temps, son logiciel pour regarder des images possède une faille ! Il ouvre la photo qui (malheur) contient un virus capable d'exploiter cette faille ! Le logiciel de visionnage d'images exécute alors le virus contenu dans l'image, qui se réplique alors dans d'autres fichiers sur son disque dur, sur le réseau, ...

La plupart des virus sont malveillants, comme les vers. Les virus sont capables de se modifier tous seuls en se répliquant pour être plus discrets. Nous n'allons pas rentrer dans les détails et expliquer le polymorphisme et autres joyeusetés, ce chapitre est juste là pour définir des termes que l'on réutilisera plus tard.

Il existe un autre type de programme, très difficile à déloger et qui est potentiellement encore plus dangereux que les autres : le **rootkit**.

### VIII.2.3. Ils s'incruster au cœur du système !

Les *rootkits*. Parlons-en. Basiquement, un *rootkit* est un programme qui va s'incruster au cœur du système d'exploitation pour devenir très difficile à déloger. Une fois installé au cœur du système, il se lance avec tous les privilèges et peut faire n'importe quoi : désactiver les systèmes de sécurité, se dupliquer dans d'autres endroits stratégiques pour devenir encore plus coriace, télécharger et exécuter des espions, etc. Il est donc particulièrement dangereux.

Un *rootkit* peut prendre n'importe quelle forme pour s'installer, et tous les *rootkits* ne sont pas détectés par les systèmes de protection (antivirus, anti-spyware...). Faites attention à ce que vous téléchargez et exécutez sur votre système et mettez à jour vos logiciels : c'est encore le meilleur moyen de ne pas avoir un système infesté de *malwares*.

## Conclusion

Ce chapitre ne servait qu'à apprendre des termes qui sont utiles pour la culture et pour savoir s'exprimer avec précision. Ces définitions sont importantes dans le milieu de la sécurité, elles permettent d'identifier clairement de quoi on parle.

## VIII.3. L'attaque de l'homme du milieu (MITM)

### Introduction

Ce chapitre de sécurité informatique sera consacré au protocole ARP. Il permet d'effectuer la correspondance adresse IP / adresse MAC, mais nous allons voir qu'il n'est pas très sûr. Une démonstration théorique de l'attaque de l'homme du milieu (couramment appelée MITM) sera présentée afin de mettre le doigt sur la sensibilité des données échangées dans un réseau local.



Le protocole ARP est utilisé avec l'IPv4, mais pas avec l'IPv6.

### VIII.3.1. Le protocole ARP ?

Tout échange de données se fait suivant un protocole, c'est-à-dire un ensemble de règles permettant d'assurer la communication entre 2 machines. Le protocole ARP, pour *Address Resolution Protocol* (protocole de résolution d'adresse), est utilisé pour établir une correspondance entre adresses IPv4 et adresses MAC. Les cartes réseau raisonnent par adresses MAC, et donc il faut leur dire à quelle adresse MAC correspond telle IP : c'est là qu'intervient l'ARP.

Admettons un réseau en étoile dont les machines sont caractérisées par leur nom, leur adresse IP, leur adresse MAC :

- Routeur, 192.168.1.1, 00:99:99:99:99:99 ;
- Ordinateur 1, 192.168.1.51, 00:11:11:11:11:11 ;
- Ordinateur 2, 192.168.1.52, 00:22:22:22:22:22.



Ces adresses MAC sont fictives et servent juste d'illustration.

Ordinateur 1 veut communiquer avec Routeur. Il connaît son adresse IP, mais la carte réseau a besoin de son adresse MAC pour lui transmettre le message. Il envoie donc une **requête ARP**. Mais pas n'importe comment ! Il va envoyer sa requête en *broadcast*. Si vous ne comprenez pas ce terme, vous devriez peut-être relire le paragraphe « Les envois de données » du chapitre sur l'adresse IPv4. Toutes les machines du réseau vont donc recevoir ce message disant « Qui est 192.168.1.1 ? Répondez-moi à 192.168.1.51 svp ! » (suivant le protocole ARP, bien entendu). Et normalement, le routeur doit lui répondre « C'est moi, mon adresse MAC est 00:99:99:99:99:99 ». Ordinateur 1 va alors stocker cette information dans une table de correspondance, la **table ARP**, au cas où il en aurait à nouveau besoin plus tard. On peut visualiser le contenu de cette table en rentrant la commande `arp -a` dans une console, sous Linux comme sous Windows. Ordinateur 1 peut maintenant communiquer avec Routeur, puisqu'il connaît son adresse IP et son adresse MAC.

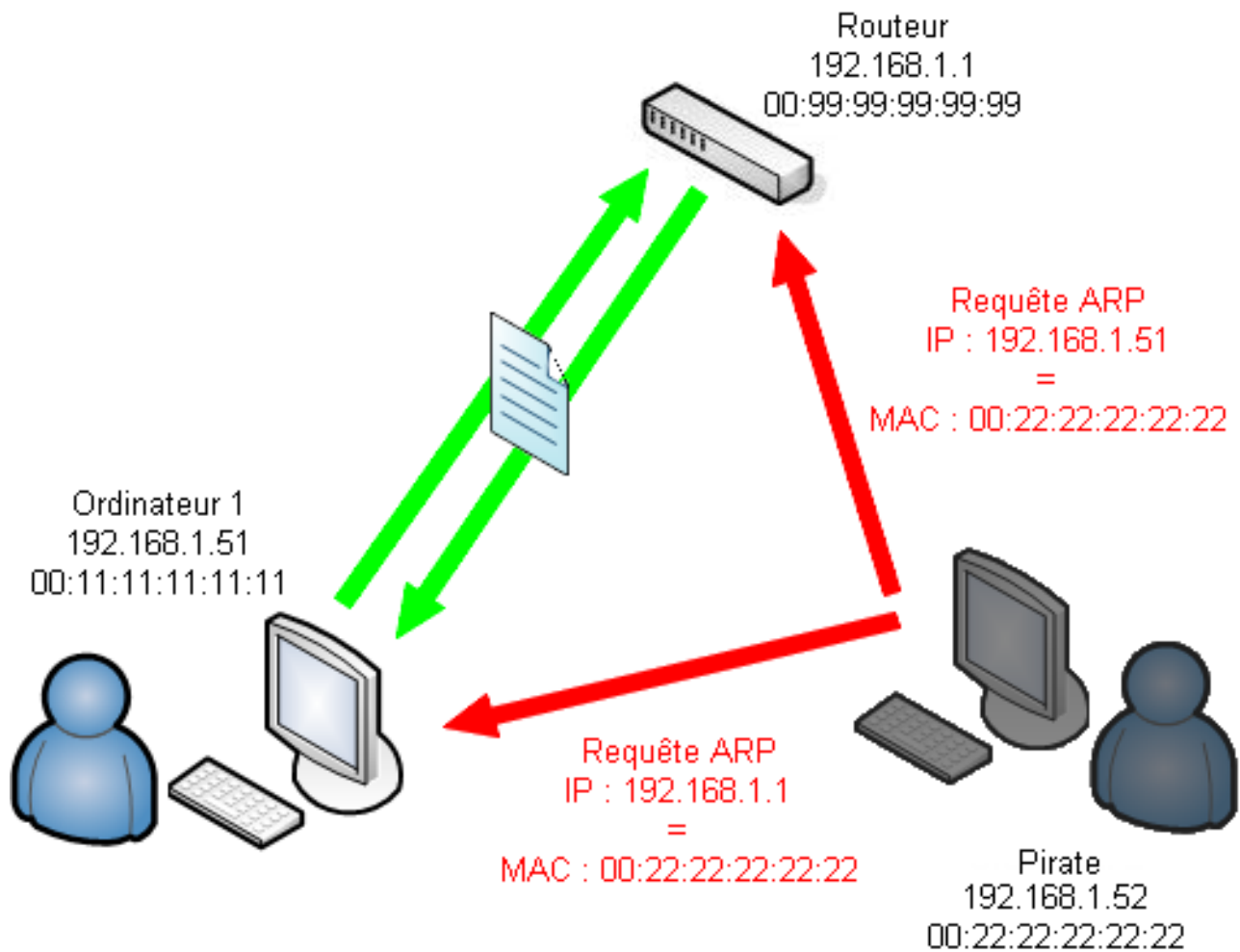
Mais...

### VIII.3.2. La faiblesse de ce protocole

Il est aisé pour une machine d'envoyer une requête ARP, aussi bien pour demander une adresse que pour répondre, pour en fournir une. Normalement, seule la machine dont l'adresse est demandée doit répondre à une requête. Mais en pratique, n'importe quelle machine peut le faire. Tout à l'heure, le routeur aurait pu répondre que son adresse MAC était 00:33:33:33:33:33, Ordinateur 1 l'aurait pris en compte sans sourciller. Simplement, quand il aurait voulu communiquer, cela n'aurait pas abouti. Pire encore ! Les ordinateurs acceptent généralement toutes les requêtes ARP leur parvenant. Ainsi, si Routeur dit à Ordinateur 1 que l'adresse MAC de Ordinateur 2 est 00:55:55:55:55:55, il va prendre en compte cette information. Cela pose un problème : un attaquant peut manipuler les tables ARP des machines présentes sur le réseau et ainsi détourner le trafic : c'est **l'attaque de l'homme du milieu**.

#### VIII.3.2.1. Le MITM: quand un intrus s'en mêle...

Restons dans le même réseau. Supposons maintenant que derrière Ordinateur 2 se trouve une personne indiscreète désireuse de savoir ce qui se dit entre Ordinateur 1 et Routeur. Il lui suffit d'envoyer une requête ARP à Ordinateur 1 disant que l'adresse MAC associée à 192.168.1.1 (l'IP du routeur) est 00:22:22:22:22:22, et d'envoyer une autre requête à Routeur disant que l'adresse MAC associée à 192.168.1.51 est 00:22:22:22:22:22. De cette manière, tout échange de données entre Ordinateur 1 et Routeur passera par Ordinateur 2 ! L'indiscreète personne peut alors analyser le trafic, puisqu'il passe par sa propre carte réseau, mais aussi l'altérer, modifier les informations qui transitent...



Les réseaux de zéro - zestedesavoir.com

FIGURE VIII.3.1. – Attaque ARP cache poisoning en cours

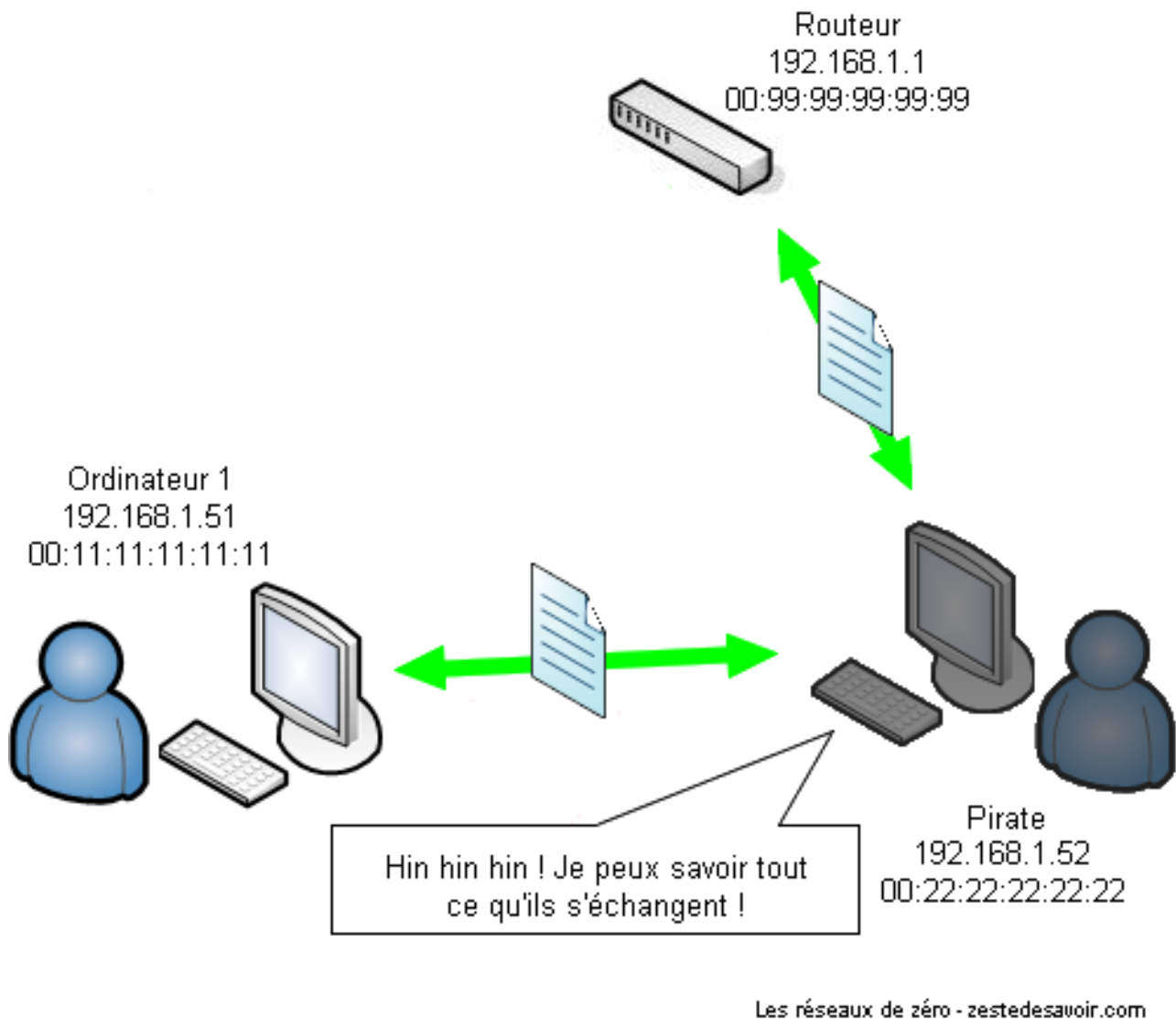


FIGURE VIII.3.2. – Attaque ARP cache poisoning

i

Plus précisément, il faut envoyer ces requêtes ARP à intervalles réguliers, car les tables peuvent se mettre à jour régulièrement. Si Ordinateur 1 et Routeur réussissent à se communiquer leurs vraies adresses MAC, le trafic reprendra normalement, et ne passera plus par Ordinateur 2.

Pour réaliser cette attaque, voici quelques logiciels utiles :

- **nemesis**, sous Linux, permet de forger des requêtes ARP, entre autres ;
- **scapy** [↗](#), sous Linux également, est un outil très pratique en réseau, il permet de faire beaucoup de choses comme forger des requêtes ou faire de l'analyse réseau par exemple ;
- **Wireshark** [↗](#), sous Linux et Windows, permet d'analyser les requêtes qui passent par votre carte réseau ;

- **Cain&Abel** [↗](#) , pour Windows, est un outil très puissant et potentiellement très dangereux, car il permet en quelques clics de notamment casser des mots de passe, de cracker des réseaux Wi-Fi et (c'est ce qui nous intéresse ici) de mettre en œuvre une attaque MITM très facilement ! Ce logiciel est génial car il est redoutable et facile d'utilisation, mais il peut être détecté comme dangereux par les antivirus. Et pour cause ! Il permet même de prendre discrètement le contrôle d'une machine à distance !

*i*

Le fait d'altérer volontairement les tables ARP s'appelle **ARP cache poisoning** (empoisonnement du cache ARP). Vous pouvez rencontrer cette expression dans le logiciel Cain&Abel. On parle de MITM uniquement quand quelqu'un s'intercale entre 2 machines pour intercepter, contrôler des données.

Si l'échange de données sensibles comme des mots de passe n'est pas chiffré, voici ce qui se passe.

### VIII.3.3. Exemple concret

Pour illustrer la théorie, voici un cas concret d'une attaque MITM réalisée sur un réseau qui nous appartient (rien d'illégal, donc). Pour ce faire, les logiciels Cain et Wireshark seront utilisés. Toutefois, Cain étant un logiciel qui permet de faire beaucoup de choses pas très honnêtes, nous ne pouvons pas vous montrer précisément les manipulations à effectuer pour réaliser un MITM.

Il y a 2 hôtes dans le réseau 192.168.1.0/24 : Pirate-PC (192.168.1.56) et Pigeon-PC (192.168.1.84). Il y a aussi un routeur qui donne accès à Internet (192.168.1.1). Pirate-PC lance une attaque MITM entre Pigeon-PC et le routeur.

## VIII. Quelques notions de sécurité réseau

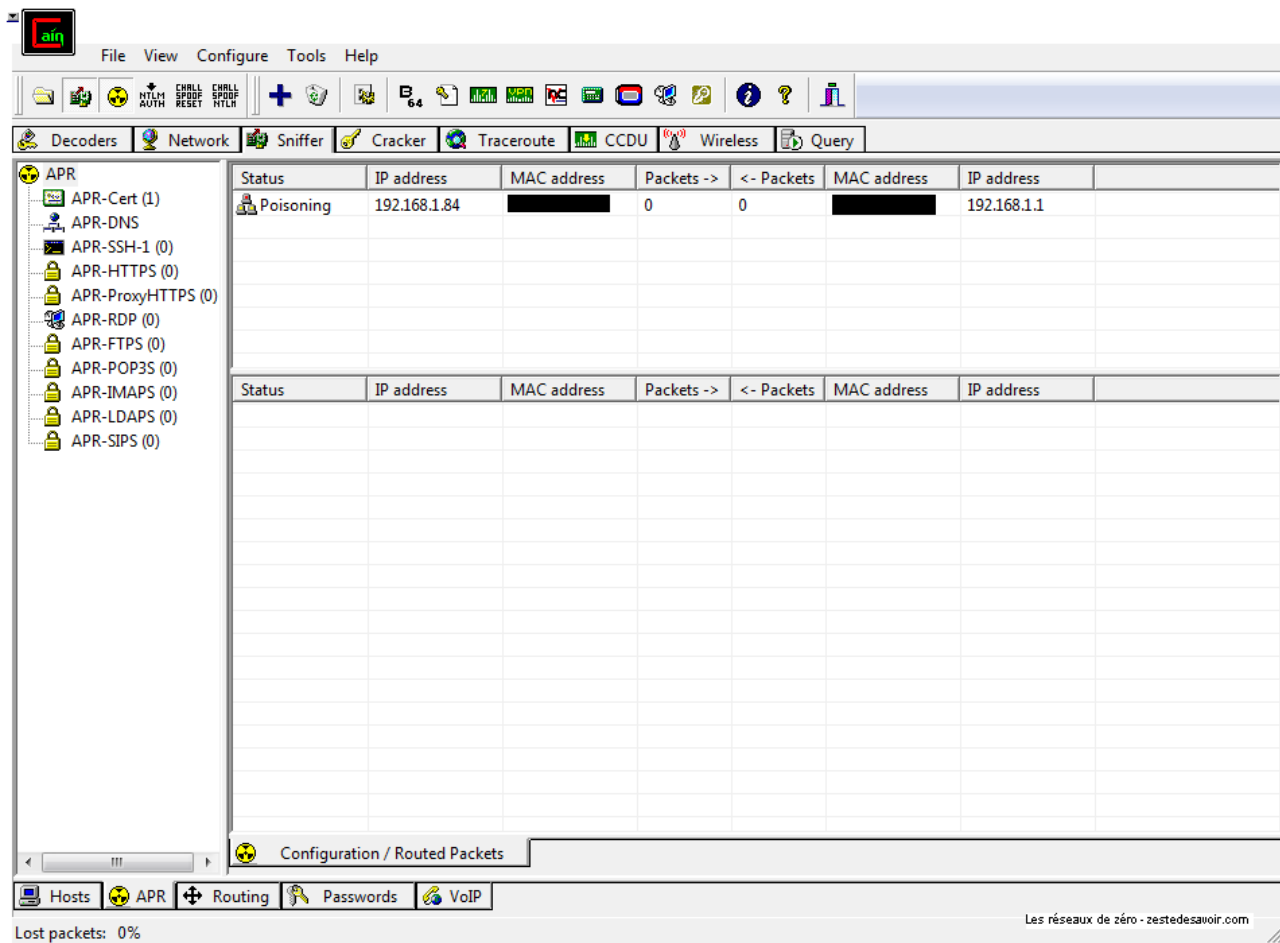


FIGURE VIII.3.3. – Interface de Cain

C'est facile, vous trouverez comment faire en cherchant un peu.

Sur Pirate-PC, on lance l'analyseur de trafic Wireshark. On lance une capture en ne demandant que les paquets en rapport avec Pigeon-PC.

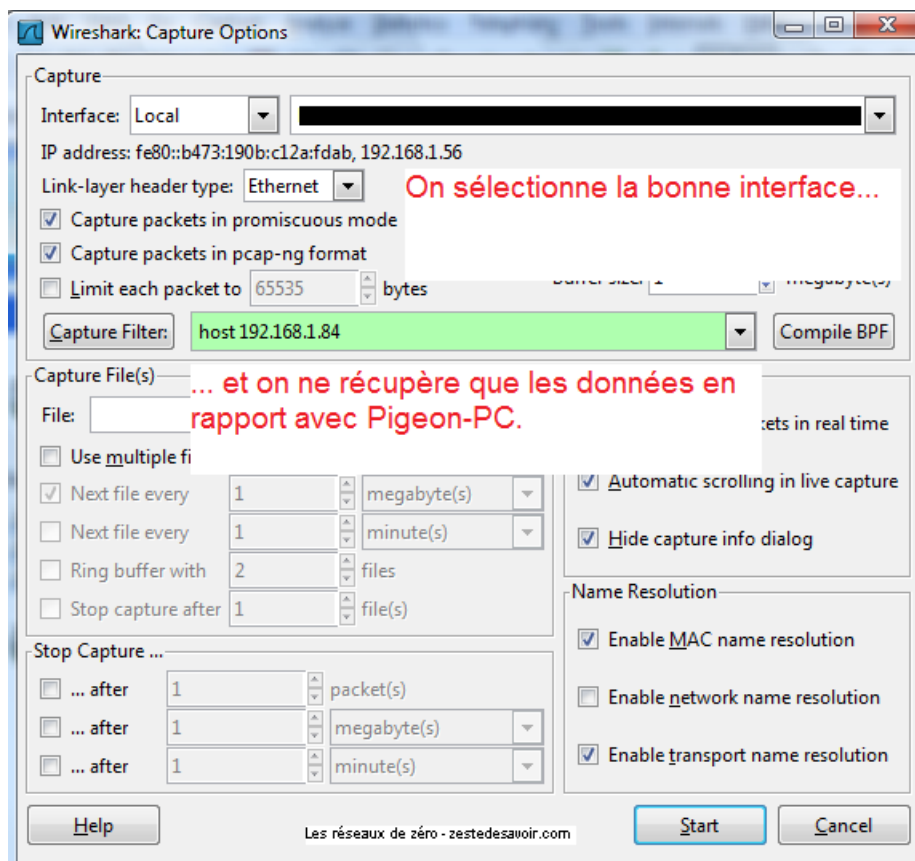


FIGURE VIII.3.4. – Options de capture dans Wireshark

*i*

Le filtre n'est pas obligatoire mais conseillé si on ne veut pas se perdre avec ses propres requêtes.

Maintenant, PigeonMan se connecte sur un site web qui ne propose pas d'identification sécurisée. On voit défiler plein de lignes incompréhensibles, mais en fouillant un peu, on tombe sur ça :

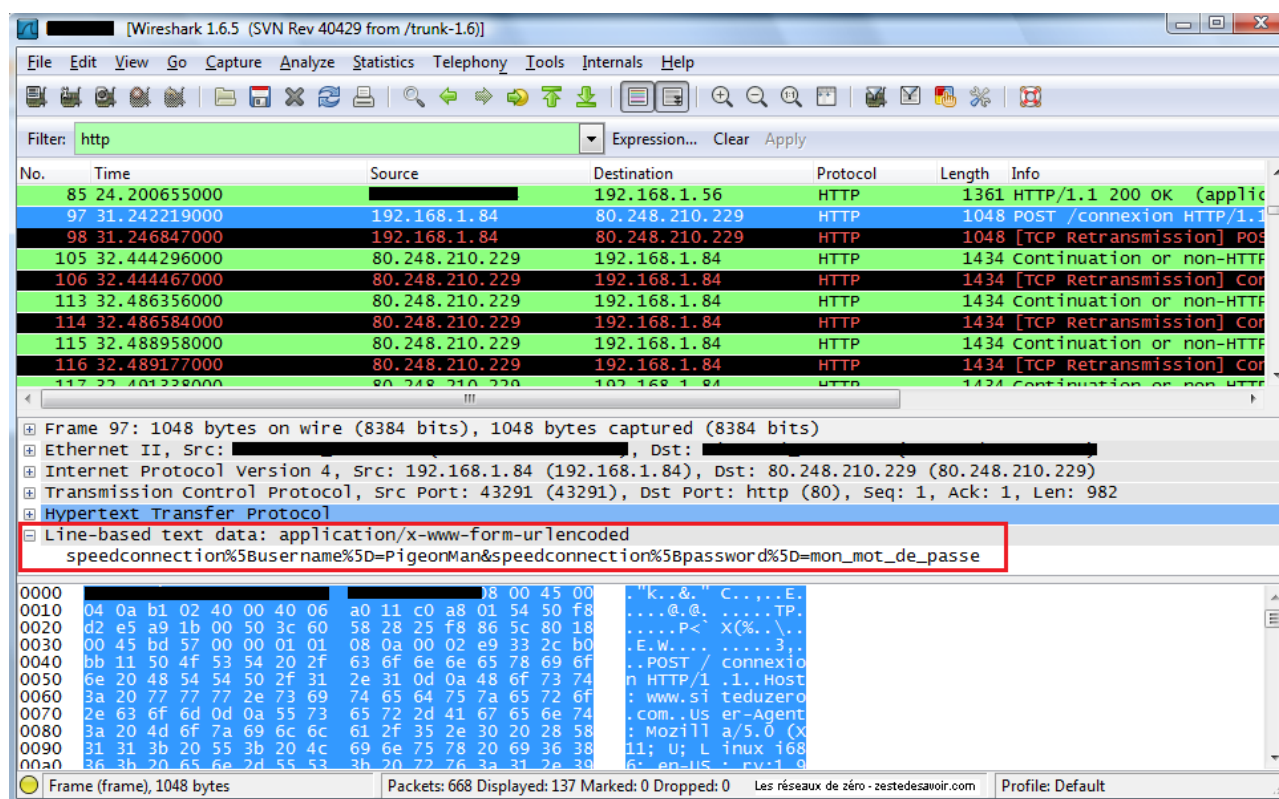


FIGURE VIII.3.5. – Paquets en clair capturés par Wireshark

Un filtre « http » a été appliqué pour qu'on ne voie que les requêtes HTTP. On voit clairement les identifiants transmis en clair.

Voilà comment on peut se faire voler ses identifiants en moins d'une minute chrono. Pour les plus fainéants, Cain propose de récupérer automatiquement les mots de passe transmis par HTTP, FTP, SMTP, etc. C'est illégal dans la plupart des pays de voler des identifiants comme cela, mais pour notre exemple, cela a été réalisé depuis un réseau qui nous appartient et avec de faux identifiants (vous ne croyez quand même pas qu'on va vous donner accès à nos comptes ? 🍌), donc ce n'est pas interdit. Mais ne comptez pas trop sur la loi pour vous protéger : pour arrêter une personne pratiquant cette attaque, il faut la prendre sur le fait, car une adresse MAC est facilement falsifiable et ne servira à rien si elle est enregistrée.

### VIII.3.4. Peut-on se défendre ?

Malheureusement, cette faiblesse du protocole ARP est difficile à combler. On peut utiliser des tables ARP statiques, mais ce n'est pas pratique et lourd à mettre en place. Sous Windows par exemple, il faut utiliser la commande `arp -s adresse_IP adresse_MAC` pour chaque adresse à fixer. Une autre solution est d'utiliser uniquement IPv6, qui n'utilise pas ARP. En entreprise, des outils comme des IDS peuvent détecter des requêtes anormales. Les *switchs* peuvent aussi être configurés pour filtrer les adresses MAC autorisées en fonction des ports.

Pour garantir l'intégrité des données que vous échangez, le plus sûr est d'utiliser des connexions sécurisées quand c'est possible, en utilisant le HTTPS dès qu'un site le permet (le navigateur Firefox depuis sa version 69 le fait automatiquement), en configurant votre client e-mail pour qu'il envoie ses requêtes *over SSL*, c'est-à-dire à travers un protocole chiffré, ... De nos jours, la plupart des logiciels chiffrent toutes les communications de bout en bout.

## VIII. Quelques notions de sécurité réseau

Il ne faut pas accepter n'importe qui sur son réseau : il convient de sécuriser son réseau Wi-Fi autant que possible (malheureusement, tous les systèmes de protection du Wi-Fi sont faillibles, même le WPA). Et même si vous utilisez un réseau câblé, si une machine du réseau est infectée par un *malware*, ce dernier pourra utiliser une attaque MITM pour espionner ce qu'il se passe sur les autres ordinateurs... 🍏 C'est pas pour vous faire devenir paranoïaque, mais c'est important de faire attention à ce qu'il se passe sur son réseau !

## Conclusion

La faiblesse étudiée est donc difficile à combler. Vous ne devriez pas laisser n'importe qui se connecter sur votre réseau. Par ailleurs, dans un réseau où les machines sont connectées en Wi-Fi non sécurisé, le simple fait d'être connecté au réseau permet de voir tout ce qui s'y passe : il suffit de configurer sa carte réseau en **mode *promiscuous***, ou en **mode *monitor*** (pas besoin d'être connecté au réseau dans ce dernier cas) et on reçoit tous les paquets, même ceux qui ne nous sont pas destinés. Même pas besoin de trafiquer les tables ARP. Le chiffrement des communications sensibles est alors essentiel.

## VIII.4. Allumez le pare-feu

### Introduction

Quand on parle de sécurité des réseaux, il y a un élément incontournable : le **pare-feu**, ou en anglais *firewall*. Cet équipement protège les machines qui sont connectées dessus, encore faut-il qu'il soit bien configuré. Dans ce chapitre, nous allons voir ses fonctionnalités principales et pourquoi il est indispensable.

#### VIII.4.1. C'est quoi ?

De manière très simplifiée, un pare-feu permet de maîtriser les flux qui passent sur le réseau. Quand on parle de pare-feu, cela peut désigner deux choses : un logiciel ou un matériel. Ils ne font pas la même chose !

##### VIII.4.1.1. Pare-feu logiciel

Il est courant d'avoir un pare-feu installé sur son ordinateur. Sous Windows par exemple, il y a de base un pare-feu activé. Il empêche l'ouverture de ports sur le système. Pour qu'un programme puisse recevoir des données sur un port, le système d'exploitation doit au préalable l'ouvrir. Ainsi, si vous exécutez un programme qui tente d'ouvrir un port, le pare-feu vous demandera si vous souhaitez l'autoriser. Bon, l'utilité est à peu près la même que de mettre des tongs pour escalader une montagne : c'est mieux que d'y aller nu pieds, mais ça ne protège de presque rien.

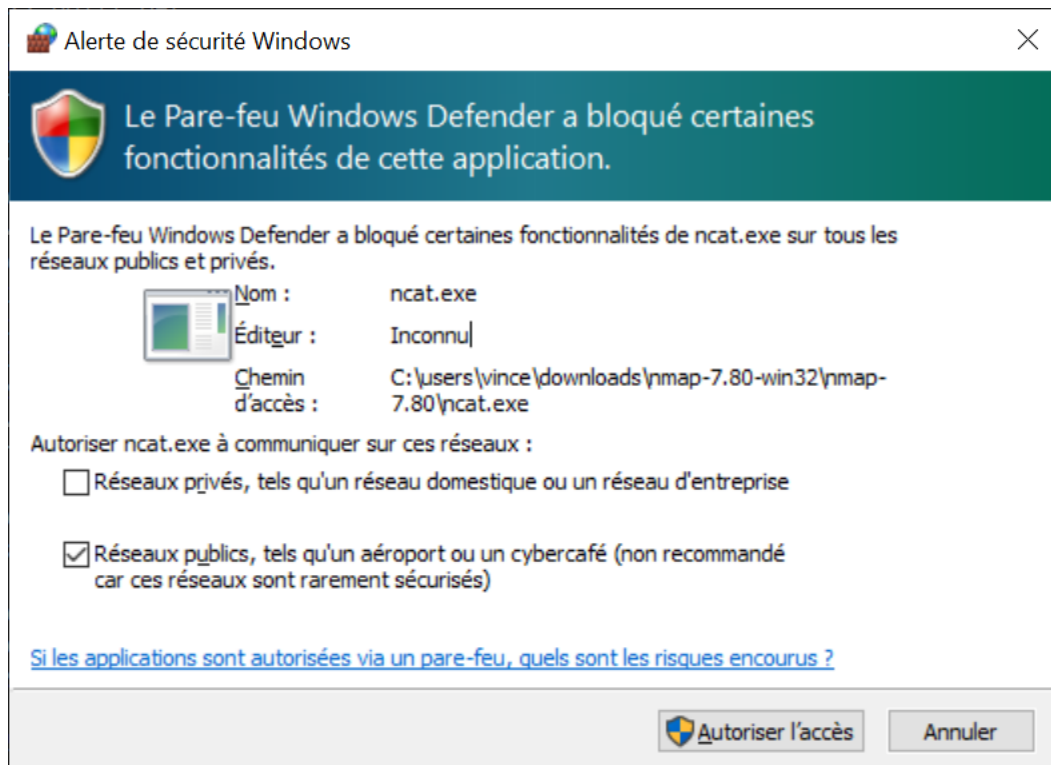


FIGURE VIII.4.1. – Demande d'ouverture de port du pare-feu Windows

Un vrai pare-feu logiciel permet de contrôler quels programmes peuvent communiquer sur le réseau et quels sont les flux autorisés. On peut citer ZoneAlarm, GlassWire ou Comodo<sup>1</sup>. Lorsqu'un logiciel veut établir une communication réseau, que ce soit en tant que serveur et donc ouvrir un port, ou en tant que client et donc initier une connexion ou envoyer des données, cela doit être autorisé par le pare-feu. Cela évite qu'un programme ne communique avec une autre machine en cachette, surtout s'il est malveillant.

#### VIII.4.1.2. Pare-feu matériel

Quand on parle de pare-feu matériel, on désigne un appareil qui ressemble à ça. L'image qui suit est publiée par Cuda-mwolfe sous licence [CC BY SA 4.0](#) (source [source](#)).



1. Ces noms sont cités à titre d'exemple. Il ne s'agit en aucun cas de recommandations.

FIGURE VIII.4.2. – Un pare-feu (CC BY SA)

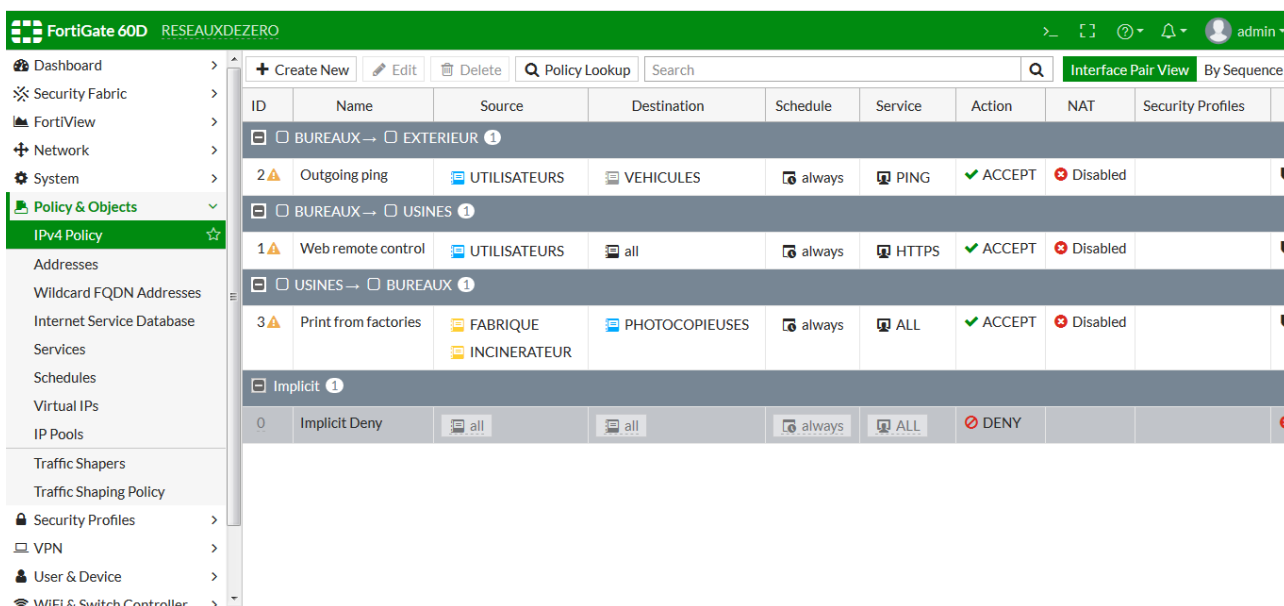
Schématiquement, il fonctionne comme un routeur, c'est-à-dire qu'il route des flux entre des réseaux. La différence, c'est qu'il dispose de nombreuses fonctionnalités permettant de contrôler, filtrer et orienter les flux.

i

La plupart des routeurs permettent aussi de maîtriser les flux, mais de manière moins précise ou moins performante. Le matériel n'est pas le même : un routeur est conçu pour router et sera performant dans cette tâche, tandis qu'un pare-feu sera plus à l'aise avec le chiffrement ou l'analyse fine de paquets.

## VIII.4.2. Sans filtre

La fonction principale d'un pare-feu est donc le filtrage des flux. Cela se fait en définissant des règles en fonction des adresses IP et des ports, à la fois pour les sources et les destinations. Visualisons sur l'interface utilisateur d'un firewall paramétré pour l'exemple ce qui est autorisé ou non. Le cas imaginé est une entreprise avec 2 usines (une fabrique et un incinérateur), des bureaux avec des postes utilisateurs et des photocopieuses, et des véhicules connectés pour circuler sur site.



ID	Name	Source	Destination	Schedule	Service	Action	NAT	Security Profiles
BUREAUX → EXTERIEUR 1								
2	Outgoing ping	UTILISATEURS	VEHICULES	always	PING	ACCEPT	Disabled	
BUREAUX → USINES 1								
1	Web remote control	UTILISATEURS	all	always	HTTPS	ACCEPT	Disabled	
USINES → BUREAUX 1								
3	Print from factories	FABRIQUE INCINERATEUR	PHOTOCOPIEUSES	always	ALL	ACCEPT	Disabled	
Implicit 1								
0	Implicit Deny	all	all	always	ALL	DENY		Enabled

FIGURE VIII.4.3. – Règles de filtrage sur un pare-feu FortiGate 60D

Il y a deux façons de concevoir une politique de filtrage :

- Tout ce qui n'est pas interdit est autorisé
- Tout ce qui n'est pas autorisé est interdit

C'est ou l'un, ou l'autre. Quand un pare-feu donne accès à Internet, cela a du sens d'autoriser tout sauf certains services particuliers, ou sauf l'accès à certains serveurs jugés dangereux (diffusion de logiciels malveillants, par exemple). C'est le premier cas. Quand on est dans un réseau d'usines où des ordres de commande sont transmis et des informations échangées entre machines et employés, il est plus sage d'interdire tous les flux sauf ceux nécessaires au travail. C'est le deuxième cas. On le rencontre souvent dans les réseaux d'entreprise. Comme vous pouvez

le voir sur l'interface précédente, le pare-feu présenté applique une politique d'interdiction par défaut.



Ce que je vois surtout, c'est qu'il n'y a pas une seule adresse IP dans cette configuration !



Quand on configure un pare-feu, on a tendance à raisonner avec des **objets**. Un objet représente un sous-réseau, un hôte ou un ensemble d'hôtes. Dans notre exemple, l'objet UTILISATEURS correspond au réseau 192.168.10.0/24. Si le réseau des utilisateurs est amené à changer, à s'agrandir ou à intégrer un autre subnet, pas besoin de tout reconfigurer : il suffit de modifier l'objet !

Name	Type	Details	Interface	Visibility	Ref.
Address 12					
FABRIQUE	Subnet	172.16.15.0/26	USINES	Visible	1
INCINERATEUR	Subnet	172.16.200.0/26	USINES	Visible	1
PHOTOCOPIEUSES	Subnet	192.168.64.0/25	BUREAUX	Visible	1
UTILISATEURS	Subnet	192.168.10.0/24	BUREAUX	Visible	2
VEHICULES	Subnet	10.30.0.0/16	EXTERIEUR	Visible	1
all	Subnet	0.0.0.0/0		Visible	1

FIGURE VIII.4.4. – Des sous-réseaux représentés sous forme d'objets

Dans une politique de filtrage, on trouve aussi des **zones**. On peut associer des interfaces physiques ou logiques, ou encore des objets à des zones. Elles correspondent à un regroupement logique d'entités, par exemple géographique. Dans notre cas, les objets PHOTOCOPIEUSES et UTILISATEURS sont dans la zone BUREAUX, car physiquement, les photocopieuses et les utilisateurs sont situés dans une zone de bureaux. Sur la capture ci-dessus, les zones sont indiquées dans la colonne "Interface". Dans la première capture, vous pouvez voir que les règles de filtrage sont découpées par blocs, comme BUREAUX > EXTERIEUR. Cela signifie que les règles de cette section ne concernent que des flux allant de la zone BUREAUX à la zone EXTERIEUR.



Sur le pare-feu de notre exemple, les objets possibles pour une règle sont limités à la zone choisie. Ainsi, quand on choisit comme source la zone "BUREAUX", on peut choisir comme objet source "UTILISATEURS" ou "PHOTOCOPIEUSES", mais pas "INCINERATEUR". Quand on choisit "all" (tout), cela correspond à tout dans la zone spécifiée. S'il n'y a pas de zone précisée, la règle s'applique à absolument tout.

Pour les ports, on raisonne aussi avec des objets. Plus précisément, dans notre cas, on parle de services. Un service est défini par un ensemble de ports de destination (et/ou source, parfois) et/ou de protocoles (TCP, ICMP, ...). Ainsi, la règle numéro 1 de notre exemple n'autorise que le service HTTPS, car on considère que les utilisateurs de la zone de bureaux ne doivent contrôler les différentes entités de la zone d'usines qu'au moyen d'une interface web et que les échanges doivent être chiffrés. La règle numéro 2 permet de *pinguer* les véhicules connectés, mais c'est tout. *A contrario*, avec la règle 3, les hôtes de la fabrique peuvent communiquer comme bon leur semble avec les photocopieuses de la zone de bureaux. Toute autre communication est interdite : les paquets ne correspondant à aucun de ces cas seront jetés, ils pourront même être *logués* (enregistrés) pour identifier les coupables et les condamner à récurer de fond en comble l'incinérateur... 🐼 Ou plus vraisemblablement, les *logs* finiront par être vidés sans jamais avoir été consultés. 🍊

Tous les pare-feux ne proposent pas une telle souplesse de paramétrage. Le modèle présenté est assez évolué. Il embarque aussi d'autres fonctionnalités bien utiles.

### VIII.4.3. Autres fonctionnalités



Certains liens de cette section ne sont accessibles qu'aux membres de Zeste de Savoir, car il s'agit de chapitres en bêta.

Notre pare-feu permet de monter [des tunnels VPN](#) 📄. Surprenamment, il ne supporte qu'IPSec, tandis que d'autres supportent aussi L2TP. Les tunnels créés génèrent des interfaces virtuelles qui servent ensuite dans les tables de routage.

À ce sujet, notre modèle supporte [les protocoles OSPF, RIP et BGP](#) 📄. 🍊

On peut aussi visualiser des statistiques détaillées du trafic reçu par le firewall en fonction d'adresses, de réseaux, de ports, s'il est autorisé ou non, etc. Cela peut servir à détecter des tentatives d'attaques ou des anomalies. Si une photocopieuse essaie de se connecter à Netflix, c'est peut-être qu'un employé s'est branché au mauvais endroit ! 🍊

Les pare-feux supportent aussi les services les plus courants. Ils peuvent donc faire office de [serveur DHCP](#) 📄, de [relai DNS](#) 📄, ils permettent la [translation d'adresses et de ports](#) 📄, etc.

Les routeurs proposés par les fournisseurs d'accès à Internet, du moins les plus courants, sont en réalité des pare-feux. Si vous cherchez dans leur interface de configuration, vous trouverez probablement certaines des fonctionnalités présentées dans ce chapitre. 🍊 Pour les particuliers, les besoins sont assez limités, protéger le réseau local derrière un NAT suffit généralement pour éviter que les ordinateurs connectés ne soient exposés sur Internet.

## Conclusion

Pour assurer la sécurité des réseaux, notamment en entreprise, les pare-feux sont indispensables. En plus de servir de routeur et de fournir des services de base, ils permettent de maîtriser finement les flux autorisés ou interdits selon les réseaux. Cela évite que des communications douteuses n'aient lieu, surtout si elles servent à des programmes malveillants.

Les pare-feux sont un monde à eux tous seuls. On en a fait une brève présentation dans ce chapitre, on peut aussi trouver une abondante littérature à leur sujet. Nous avons vu un modèle en particulier, le Fortigate 60D, du constructeur Fortinet. Ce dernier propose une gamme de

### *VIII. Quelques notions de sécurité réseau*

firewalls adaptés à différents besoins. Parmi les autres, on peut citer Juniper, Checkpoint ou encore Cisco.

# Conclusion

Nous espérons que ces quelques chapitres vous auront apporté un peu de culture et de savoir sur la sécurité des réseaux.

## Neuvième partie

### Annexes

# Introduction

Cette partie contient des chapitres traitant de notions utilisées en réseau, sans en être fondamentalement. Ils peuvent être consultés indépendamment du reste du tuto.

## IX.1. Binaire et hexadécimal : partez sur de bonnes bases !

### Introduction

Dans la plupart des domaines de l'informatique, il est indispensable de connaître le binaire. Le réseau ne fait pas exception ! Nous allons d'abord voir de quoi il s'agit et pourquoi cela sert en réseau. Nous en profiterons pour ensuite aborder l'hexadécimal, puis nous concluons avec un point sur le stockage des nombres par les ordinateurs.

#### IX.1.1. Décimal vs binaire : un peu de pratique

Comme nous sommes dans un cours de réseau, nous allons d'abord nous familiariser avec la conversion d'une adresse IP de sa notation binaire à sa notation décimale et vice versa. 🍊

?

Ça a une utilité ou c'est juste pour faire fonctionner nos neurones ?

Eh bien, ça va nous préparer à faire des calculs intéressants, comme la personnalisation des masques de sous-réseaux ou la détection d'erreurs de transmission.

##### IX.1.1.1. Système décimal

Le système de numération décimale est le plus connu aujourd'hui. Oui, il s'agit bien de la fameuse suite des chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Ce système utilise la base 10.

Bon, ne trouvant vraiment pas quoi dire de plus sur cette technique de numération, nous allons nous contenter de citer Wikipédia pour les avantages et inconvénients de ce système (uniquement utile pour la culture hein, ce n'est pas à apprendre par cœur 🍊 ) :

La base dix comporte quelques atouts :

- Le compte sur les dix doigts est très intuitif
  - Elle est construite sur un nombre pair, et la division par deux est la plus courante
  - Son ordre de grandeur est satisfaisant, car il permet de réduire considérablement la longueur d'un grand nombre par rapport à une base deux, tout en conservant des tableaux d'additions et de multiplications mémorisables
  - Les normes internationales sont construites sur cette base
  - Elle est la plus courante
- Cependant, la base dix n'est pas celle qui offre le meilleur bénéfice, car elle ne s'appuie pas sur un nombre ayant des propriétés avantageuses :
- Un nombre comportant beaucoup de diviseurs (ex. : 12 ou 60) aurait eu un aspect pratique, or 10 n'en a que deux (2 et 5), et la division par cinq n'est pas la plus

- courante
- Un nombre premier serait intéressant pour les mathématiques, car, dans une telle base, les nombres à virgule s'écriraient aisément sous forme de fraction irréductible, or dix n'est pas premier
- Une puissance de deux serait adaptée à l'informatique, or dix n'est pas puissance de deux

Wikipédia

### IX.1.1.2. Système binaire

Le système binaire est un système de numération, tout comme l'est le système décimal, sauf que le système binaire utilise une base de 2. Il n'y a que deux chiffres : 0 et 1. Ces chiffres qui composent les nombres binaires sont appelés *bits* (combinaison des deux mots anglais *BI*nary et *digi*T). Si vous vous rappelez bien, quand on a huit *bits*, on a un octet. 🍊

Le système binaire est le système de base des calculs informatiques : votre processeur travaille sur une suite de 0 et 1, par exemple pour faire des vérifications. Si telle partie a du courant, le *bit* vaudra 1, sinon le *bit* vaudra 0.

Donc, les adresses IP et les masques de sous-réseaux que vous avez pu voir dans ce cours ne sont qu'une suite de 0 et 1. Et nous allons dans ce chapitre apprendre à faire la conversion d'une adresse IP écrite en binaire à une adresse IP écrite en décimal et vice versa.

i

Le système binaire est assez complexe quand il s'agit de représenter des chiffres négatifs, mais c'est important quand on veut faire de la programmation de bas niveau avec un langage comme l'assembleur, vous pouvez donc lire [cet article](#) ↗. Prenez bien en compte que la lecture n'est pas obligatoire, ce que nous allons vous dire dans cette sous-partie suffira pour continuer la lecture de ce tutoriel.

### IX.1.1.3. C'est parti, touchons du binaire

Bon, avant tout, nous allons faire un petit rappel. Une adresse IP est une suite de 4 groupes de chiffres séparés par un point. Chaque « groupe » vaut 1 octet, soit 8 *bits*. Ce qui nous intéresse ici est le *bit*.

D'après les explications ci-dessus, un *bit* est un nombre qui ne peut prendre que 2 valeurs possibles : 0 ou 1. Comme nous avons 8 *bits*, nous allons donc représenter une adresse IP par 32 chiffres qui ne seront que des 0 et 1. En effet, chaque portion de l'adresse IP valant 8 *bits*, nous aurons 8 chiffres par portion. Et comme une adresse IP a 4 portions, nous aurons donc  $8 * 4$  chiffres, soit 32 chiffres dans la représentation binaire d'une adresse IP. 🍊

Nous allons prendre une adresse simple pour commencer : 12.3.2.1.

Nous allons écrire cette adresse IP en binaire. Rappelez-vous, chaque portion (délimitée par un point) de l'adresse IP vaut 8 *bits*. Donc vous devez écrire 4 suites de 8 chiffres séparées par des points. Pour commencer, on va mettre tous ces chiffres à 0, pour voir pas à pas comment on procède. 🍊

Nous avons donc :

```
0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0
```

Voilà, nous y sommes presque. Nous avons écrit 32 chiffres, donc 4 groupes de 8 chiffres.

Chacun de ces chiffres ne doit valoir que 1 ou 0. Ici nous n'avons qu'une suite de zéros, cela veut dire que sa valeur correspondante en décimal serait 0.0.0.0. Mais nous voulons représenter

## 12.3.2.1, alors comment faire ?

Dans un nombre en binaire, chaque *bit*, s'il vaut 1, correspond à un certain nombre en décimal. Ainsi, de la droite vers la gauche on a :

- chiffre 1 : 1
- chiffre 2 : 2
- chiffre 3 : 4
- chiffre 4 : 8
- chiffre 5 : 16
- chiffre 6 : 32
- chiffre 7 : 64
- chiffre 8 : 128

i

**Point vocabulaire important** : si un *bit* vaut 1, on dit qu'il est allumé, sinon c'est qu'il est... éteint, bingo ! 🍊

Ainsi, dans un nombre en binaire, pour avoir sa correspondance en décimal, il suffit de regarder quels *bits* sont allumés (toujours de la droite vers la gauche), de noter leur correspondance en décimal et de faire la somme des correspondances et hop, vous avez le nombre en décimal ! Avouez que c'est moins compliqué que ça en avait l'air tout à l'heure. 🍊

Pour passer d'un nombre décimal à son équivalent en binaire, il faut décomposer le nombre décimal en une somme faisant intervenir les nombres correspondants à certains *bits* allumés (en fonction des possibilités) et ensuite, d'allumer les *bits* dont les correspondances sont utilisées.

On va tout de suite prendre un exemple avec notre conversion de 12.3.2.1 en binaire. Commençons par 12. Décomposons-le à partir des correspondances données ci-dessus :  $12 = 8 + 4$ . Il nous faut donc allumer le *bit* dont la correspondance vaut 8 et celui dont la correspondance vaut 4 soit le 3ème et le 4ème en allant de la droite vers la gauche. Ce qui nous donne : 0 0 0 0 1 1 0 0. Et voilà, vous venez de représenter 12 en binaire. Faites pareil pour les 3 autres octets (ce n'est vraiment pas dur 🍊).

Ceci nous donne au final :

- 1er octet : 00001100
- 2ème octet : 00000011
- 3ème octet : 00000010
- 4ème octet : 00000001

Donc, l'adresse 12.3.2.1 en binaire c'est : 00001100 . 00000011 . 00000010 . 00000001

i

Connaître les puissances de deux vous facilitera vraiment la conversion. Avec un peu de pratique, ça deviendra automatique.

Voici donc le tableau récapitulatif sur un octet de la correspondance des *bits* :

Valeur décimale	128	64	32	16	8	4	2	1

Puis- sance de 2	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Exemple : $87_{10}$	1	0	1	0	1	1	1	1

Jusqu'à présent, nous avons travaillé avec des nombres allant jusqu'à 255. Comment faire pour représenter en binaire des nombres plus grands ? Eh bien, il suffit d'étendre le tableau ci-dessus pour s'en rendre compte.

Valeur décimale	1024	512	256	128	64	32	16	8	4	2	1
Puis- sance de 2	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

Ainsi, si on souhaite représenter le nombre 1337, il suffit de le décomposer en puissances de deux :

$$1337 = 1024 + 256 + 32 + 16 + 8 + 1$$

$$1337 = 2^{10} + 2^8 + 2^5 + 2^4 + 2^3 + 2^0$$

On peut donc écrire  $1337_{10} = 10100111001_2$  !



La notation  $\text{NOMBRE}_{(\text{BASE})}$  est très courante et permet de préciser la base mathématique du nombre. Si on ne précise pas la base, c'est du décimal (base 10). Attention, 10 (décimal) est différent de  $10_{(2)}$  (binaire, ce qui fait 2 en décimal).

Pour terminer cette section, voici une astuce pour la conversion décimal - binaire qui vous facilitera la tâche, le temps de s'habituer. Commencez par écrire un tableau comme celui ci-dessus. Regardez la colonne la plus à gauche et si votre nombre est supérieur ou égal à cette puissance de 2, retranchez cette valeur à votre nombre et inscrivez 1 en dessous. Sinon, inscrivez 0. Continuez avec la colonne suivante jusqu'à arriver au bout.

Voici un exemple d'application de cette méthode pour écrire le nombre 30 en binaire. On prend 6 *bits* pour avoir suffisamment de place.

Valeur décimale	32	16	8	4	2	1
Puis- sance de 2	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Nombre au dé- part : $30_{10}$	?	?	?	?	?	?

30 n'est pas plus grand que 32, on continue	0	?	?	?	?	?
30 est supérieur à 16, on fait $30 - 16 = 14$	0	1	?	?	?	?
14 est supérieur à 8, on fait $14 - 8 = 6$	0	1	1	?	?	?
6 est supérieur à 4, on fait $6 - 4 = 2$	0	1	1	1	?	?
2 est égal à 2, on fait $2 - 2 = 0$	0	1	1	1	1	?
0 est inférieur à 1, on inscrit un 0	0	1	1	1	1	0

On a ainsi  $30_{10} = 011110_2$ .

*i*

On a ici écrit la valeur sur 6 *bits*, mais on aurait aussi pu le faire sur 5 *bits* ou plus. Rajouter un zéro devant un nombre, quel qu'il soit, ne change pas sa valeur. Comme on travaille souvent avec des octets, donc des groupes de 8 *bits*, il est fréquent qu'on écrive les valeurs en binaire sur 8 chiffres en rajoutant des zéros inutiles. Cela ne change rien aux valeurs :  $011110_2 = 11110_2 = 00011110_2$ .

### IX.1.2. Un point sur l'hexadécimal

Vous avez réussi à appréhender le binaire pour, par exemple, calculer les masques en IPv4 ? Nous allons donc partir du binaire pour expliquer l'**hexadécimal**. Comme son nom l'indique, il s'agit de la base 16 (hexa : six, décimal : dix). Elle sert notamment à la notation des adresses IPv6.

En binaire, il y a 2 chiffres : le 0 et le 1. Pour représenter un nombre plus grand que 1, on rajoute un ou plusieurs chiffres. En décimal, c'est pareil : on écrit plusieurs chiffres pour représenter un

nombre plus grand que 9. En hexadécimal, c'est pareil, mais on a 16 chiffres. Pour représenter un nombre plus grand que 15, on utilisera plus de chiffres.

Voici la correspondance entre nombres binaires, décimaux et hexadécimaux :

Décimal	Binaire	Hexadé- mal
0	0	0
1	1	1
2	10	2
...	...	...
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
...	...	...

Les chiffres hexadécimaux vont donc de 0 à F : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Nous ne nous intéresserons pas directement à la conversion hexadécimal - décimal, mais rien ne vous empêche de chercher par vous-mêmes. 🍊 Seule la conversion hexadécimal - binaire (et inversement) sera expliquée ici. Et c'est tout simple ! 🍊

Un chiffre hexadécimal se représente par 4 chiffres binaires. Complétons le tableau précédent en écrivant les zéros inutiles pour les nombres binaires.

Hexadé- mal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Maintenant, pour convertir un nombre hexadécimal en binaire ou inversement, il vous suffira de vous reporter à ce tableau ! À vrai dire, on le retient par cœur assez rapidement, avec un peu d'entraînement.

Avec IPv6, on utilise souvent le nombre FE80<sub>(16)</sub>. Qu'est-ce que ça donne en binaire ?

👁 Contenu masqué n°7

Comme un chiffre hexadécimal correspond à 4 chiffres binaires, on sépare généralement les blocs de 4 bits pour y voir plus clair.

Et voilà ! 🧙

## Conclusion

Avant de clore ce chapitre, sachez que les ordinateurs ne peuvent traiter des valeurs numériques que sous forme d'octets. Le nombre 98<sub>10</sub> peut ainsi être stocké sur un octet, car il faut 7 *bits* pour le représenter. En revanche, le nombre 256<sub>10</sub> nécessite 9 *bits*, on doit alors nécessairement mobiliser 2 octets !

Désormais, vous êtes en mesure de convertir des nombres du binaire au décimal et à l'hexadécimal ! Outre la compréhension du *subnetting*, cela vous sera utile pour comprendre les systèmes de détection d'erreurs de transmission, que vous découvrirez dans une prochaine annexe.

## Contenu masqué

### Contenu masqué n°7

Il suffit de lire le tableau et de remplacer chaque chiffre hexadécimal par sa valeur décimale !  
 FE80<sub>(16)</sub> = 1111 1110 1000 0000<sub>(2)</sub>

[Retourner au texte.](#)

## IX.2. La détection d'erreurs avec la somme de contrôle

### Introduction

Nous avons introduit le principe de la somme de contrôle dans le chapitre « [Aujourd'hui, contrôle !](#) ». Si vous êtes curieux, nous allons explorer un peu le côté technique de ce mécanisme dont la vocation est, comme l'indique le titre, la détection d'erreurs. Enjoy ! 🍊

#### IX.2.1. La vérification de parité

Bienvenue à votre premier baptême du feu. 🍊

Comme l'indique le titre, nous allons débiter notre exploration des algorithmes de computation de somme de contrôle en étudiant **l'algorithme de vérification longitudinale de parité**.

?

La parité, c'est quand il y a autant d'hommes que de femmes ? 🍊

La parité est la propriété de tout nombre dont le reste d'une division par 2 est un nombre entier. En d'autres termes, on appelle nombre pair tout nombre qui est multiple de deux. Tout le monde ici sait qu'est-ce qu'un nombre pair et impair, quand même ? 🍊

Un algorithme de vérification longitudinale de parité se base sur la parité. Mais pas la parité des nombres entiers : il se base sur la parité de *bits*.

?

Des *bits* ? Des 0 et 1 pairs ? 🍊

D'abord on se calme, n'ayez pas peur. 🍊 La parité de *bit* ou le *bit* de parité, c'est la forme la plus simple de détection d'erreur. Ne commencez pas tout de suite à paniquer, l'algorithme de vérification de parité est plus simple à comprendre que les autres à venir. 🍊

Imaginez que vous avez une suite de *bits* 10011001. Cette suite de *bits* a 4 *bits* allumés (4 *bits* qui valent 1). Pour nous les humains, nous savons par expérience que 4 est un nombre pair, nous n'avons pas besoin de faire des calculs pour vérifier la parité de ce nombre. Mais pour un ordinateur qui fonctionne avec des 0 et 1, comment savoir que 4 est pair ? Comment savoir que l'ensemble des *bits* allumés dans une suite de *bits* est un nombre pair ? On utilise pour ce faire un *bit* de parité.

En toute logique, il y a deux types de *bit* de parité : le *bit* de parité pair et le *bit* de parité impair.

?

Ça ne nous dit pas ce qu'est un *bit* de parité...

Un *bit* de parité est un *bit* ( si si 🍊 ) que l'on ajoute au début d'une suite de *bits* pour nous informer de la parité du nombre de *bits* allumés. C'est un peu difficile à comprendre, nous vous l'accordons, mais nous sommes là pour vous simplifier cela au maximum. 🍊  
 Prenons une suite de *bits* **1001111**. Cette suite de *bits* a 5 *bits* allumés, c'est-à-dire 5 *bits* qui valent 1. Nous allons ajouter un *bit* au début de cette suite. Ce *bit* va déterminer si 5 est pair ou impair. Ce *bit* ajouté, c'est cela le *bit* de parité. 🍊

?

C'est quoi la différence entre le *bit* parité pair et le *bit* de parité impair ?

Bonne question ! Chacun de ces deux *bits* sert à déterminer la parité d'un nombre, c'est-à-dire déterminer si le nombre de *bits* allumés est un nombre pair ou impair. Mais quelle est la différence entre ces deux *bits* ?

#### IX.2.1.1. Le bit de parité pair

Bien. Quand vous utilisez un *bit* de parité pair, le *bit* de parité qui sera ajouté à votre suite de *bits* aura pour valeur 1 si le nombre de *bits* allumés est impair.

Un exemple ? Prenons la suite **1111111**. Nous avons 7 *bits* et 7 est un nombre impair. Le *bit* de parité sera donc 1. 🍊

Retenez ceci : **l'ajout du *bit* de parité pair rendra l'ensemble des *bits* allumés pair.** 7 est un nombre impair, n'est-ce pas ? Notre *bit* de parité pair doit donc valoir 1. Et si nous ajoutons 7 *bits* allumés + le *bit* de parité pair, ça donne 8 *bits* et 8 est un nombre pair. Donc ajouter un *bit* de parité pair rendra toujours l'ensemble des *bits* allumés pair. 🍊

#### IX.2.1.2. Le bit de parité impair

Si vous utilisez un *bit* de parité impair, le *bit* doit valoir 1 si l'ensemble des *bits* allumés est pair.

Un exemple ? **1101100**. Cette suite a 4 *bits* allumés et 4 est un nombre pair, donc le *bit* de parité impair que l'on ajoutera à cette suite vaudra 1. Contrairement au *bit* de parité pair, celui-ci rendra l'ensemble des *bits* impair. 4 est pair, si on ajoute un *bit* de parité impair (qui vaut 1) on se retrouve donc avec 5 *bits* allumés, et 5 est un nombre impair. 🍊

En résumé, retenez la règle suivante :

**Soit S, une suite de *bits* composée de n *bits* allumés. Un *bit* de parité pair aura pour valeur 0 si n est un nombre pair. Un *bit* de parité impair aura pour valeur 0, si n est un nombre impair.**

Il est vraiment important de comprendre le principe du *bit* de parité (pair ou impair), car c'est le principe de base de l'algorithme de vérification longitudinale de parité. C'est d'ailleurs important pour comprendre le contrôle de redondance cyclique.

Nous allons donc faire quelques exercices : 3 exercices pour le *bit* de parité pair et 3 autres pour le *bit* de parité impair. C'est parti ! :pirate :

### IX.2.1.3. Exercices sur le bit de parité pair

Commençons par quelque chose de très simple : **1110000**. Voilà une suite de *bits* qui a 3 *bits* allumés. Nous voulons utiliser un *bit* de parité pair. Nous savons que le *bit* de parité pair vaut 1 si et seulement si le nombre de *bits* allumés est un nombre impair. Ça tombe bien parce que 3 est impair. Donc nous allons réécrire cette suite de *bits* en intégrant le *bit* de parité pair. Le *bit* de parité pair rendra le nombre de *bits* allumés pair comme nous l'avons vu. Le *bit* de parité se place au début de la suite de *bits*.

**11110000**

Voilà, c'est fait. 🍊 Nous avons désormais 4 *bits* allumés, ce qui est un nombre pair, car l'ajout du *bit* de parité pair rendra toujours l'ensemble des *bits* allumés pair. 🍊

Deuxième exercice !

Soit la suite de *bits* **1111100**. Nous avons 5 *bits* allumés. Notre *bit* de parité pair devra donc valoir 1 puisque 5 est impair. 🍊

**11111100**

Nous avons désormais 6 *bits* allumés et 6 est un nombre pair.

Un dernier exercice pour la route !

Soit la suite **1100000**. 2 est un nombre pair. Or le *bit* de parité pair prend la valeur 1 si et seulement si le nombre de *bits* allumés est impair. Il s'avère que 2 n'est pas impair, donc le *bit* de parité ne peut pas être allumé. Il aura pour valeur 0.

Nous avons fait exprès de vous faire voir ce cas. Si le nombre de *bits* allumés est un nombre pair, il vaudra 0, sinon il vaudra 1 comme nous l'avons fait pour les premiers exercices. Allez, écrivons cette suite avec le *bit* de parité.

**01100000**

Remarquez que  $n$  (le nombre de *bit* allumés, 2 en l'occurrence) + 0 (le *bit* de parité pair) nous donne toujours un nombre pair pour respecter la règle. 🍊

### IX.2.1.4. Exercices sur le bit de parité impair

Vous vous souvenez de la règle ? Le *bit* de parité impair vaut 1 lorsque le nombre de *bits* allumés est un nombre pair, sinon il vaut 0. Cela veut dire que si le nombre de *bits* allumés est un nombre impair, le *bit* de parité impair vaudra 0.

Sans plus tarder, commençons.

Soit la suite **1100000**. Cette suite a 2 *bits* allumés. 2 est un nombre pair, donc le *bit* de parité impair vaudra 1. Écrivons cela :

**11100000**

Remarquez que nous avons désormais 3 *bits* allumés et 3 est un nombre impair. Ça confirme bien la règle : l'ajout d'un *bit* de parité impair rendra impair le nombre de *bits* allumés. 🍊

Continuons !

Soit la suite **1010100**. Nous avons 3 *bits* allumés. 3 est impair, par conséquent le *bit* de parité vaudra 0. Cela nous donne :

**01010100**.

Pour terminer, considérons la suite **1111110**. 6 est un nombre pair, donc le *bit* de parité impair vaudra 1. Nous sommes des pros maintenant, ça devrait couler de source :

**11111110**

### IX.2.1.5. Conclusion

Le principe des *bits* de parité sert de base à d'autres opérations plus complexes. Ce système est aussi utilisé tel quel dans certains protocoles de niveau 2, nous aurons certainement l'occasion d'en reparler.

Pour conclure, il y a 4 règles d'or à retenir :

- Un *bit* de parité pair vaut 1 lorsque le nombre de *bits* allumés dans une suite est un nombre impair. Sinon il vaut 0.
- L'ajout d'un *bit* de parité pair à une suite donnera un nombre pair de *bits* allumés.
- Un *bit* de parité impair vaut 1 lorsque le nombre de *bits* allumés dans une suite est un nombre pair. Sinon il vaut 0.
- L'ajout d'un *bit* de parité impair à une suite donnera un nombre impair de *bits* allumés.

### IX.2.2. La somme de contrôle de Fletcher

Le *bit* de parité, c'est assez simple, c'est rapide à calculer, mais ce n'est pas très performant. Ce serait mieux d'avoir une information plus fiable. D'un autre côté, on ne veut pas non plus avoir des méthodes qui sont coûteuses en ressources et en temps.

Jusqu'au début des années 1980, un système de somme de contrôle assez simple était utilisé. Il consiste à découper le message à transmettre en blocs de taille fixe (1 octet, 2 octets, 4 octets, ...). On fait ensuite la somme des valeurs de chacun de ces blocs.

Prenons pour exemple le mot « Bonjour », encodé en **ASCII**<sup>1</sup>. Découpons-le en blocs de 1 octet. On obtient les valeurs suivantes :

B	o	n	j	o	u	r
066	111	110	106	111	117	114

La somme de toutes ces valeurs donne...

Quelqu'un aurait une calculatrice ? 🍌

735. On ajoute cette

valeur au bout du message.

?

Mais comment le destinataire sait quelle partie du message correspond à la somme de contrôle ?

Ça, c'est défini en amont par le protocole utilisé ! 🍌 Dans notre exemple, on va supposer que seul le dernier octet correspond à la somme de contrôle.

?

Ça ne peut pas aller ! On ne peut pas stocker la valeur 735 sur un seul octet, il en faut 2 !

C'est exact. On commence à mettre le doigt sur une importante problématique. Même si c'était deux octets, pour peu que le message soit long, on pourrait dépasser la valeur maximale de 65535. Alors comment faire ?

C'est là qu'intervient **l'arithmétique modulaire**. Ce concept mathématique introduit la fonction **modulo**. Le modulo, c'est une opération qui consiste à conserver le reste d'une division entre deux nombres entiers. On note cette opération **mod** ou, dans certains langages de programmation, **%**. Ainsi, **10 mod 3 = 1**, car le reste de la division de 10 par 3 est 1. Essayez de résoudre ces opérations de tête pour vous entraîner :

- $13 \bmod 4 = ?$
- $50 \bmod 10 = ?$
- $735 \bmod 256 = ?$

👁 Contenu masqué n°8

?

Quel rapport avec la somme de contrôle ?

Regardez la dernière opération que vous avez calculée. Vous avez réussi à faire rentrer la valeur 735 sur un octet ! 🍌 Bon, pas tout à fait. À partir de la somme calculée, nous avons obtenu une valeur que nous n'aurions pas obtenue avec un autre message. S'il y avait eu une erreur de transmission, par exemple sur le dernier *bit* de la 4ème lettre du message, le « j » serait devenu un « k », et la valeur numérique décimale serait 107 et non 106. La somme des valeurs serait 736 et non 735. L'opération modulo aurait été  $736 \bmod 256 = 224$ , et non 223. Le récepteur aurait donc calculé une somme de contrôle différente de celle fournie par l'expéditeur, c'est-à-dire 223, qui a été jointe au message d'origine. On sait donc qu'il y a forcément eu une erreur de transmission.

Cette méthode est pratique, car elle est assez simple et rapide à calculer. Toutefois, elle n'est pas très fiable : on peut tomber sur le bon résultat malgré des erreurs de transmission. C'est ce qu'on appelle une **collision**. Pire encore : on tombera quand même sur la bonne somme de contrôle avec les blocs (dans notre exemple, les lettres) inversées.

i

Nous avons choisi le diviseur 256 dans notre exemple. Cela permet de s'assurer que la somme de contrôle rentrera sur un octet. En effet, tout nombre modulo 256 ne peut pas donner un résultat supérieur à 255, qui est la valeur maximale stockable sur un octet. On peut aussi choisir un diviseur inférieur, cela fonctionnerait de la même manière. Par contre, les chances de collision seront supérieures, puisque le nombre total de valeurs possibles sera réduit. Si nous avons choisi de stocker notre somme de contrôle sur 2 octets, nous aurions pu choisir un diviseur allant jusqu'à 65536.

### IX.2.2.1. Un Fletcher sauvage apparaît !

Vers la fin des années 1970, le physicien John Fletcher, du laboratoire national de Lawrence Livermore en Californie, propose une amélioration à ce système. Il consiste à ajouter, en plus de la somme de contrôle étudiée précédemment, la somme de toutes les étapes intermédiaires. Ainsi, cela permet de prévenir l'inversion de blocs lors de la transmission et réduit le risque de collision. Comme c'est difficile à visualiser, faisons tout de suite un exemple.

Reprenons le message sur lequel nous avons travaillé. Nous allons calculer, étape par étape, deux sommes de contrôle : celle d'origine, et celle de Fletcher, qui consiste à additionner la première *checksum* à chaque étape. Nous conservons la même taille de bloc et le même modulo, que nous appliquerons à chaque étape pour plus de lisibilité (cela ne change rien que cette opération soit faite à la fin ou au fur et à mesure).

B	o	n	j	o	u	r
---	---	---	---	---	---	---

066	111	110	106	111	117	114
-----	-----	-----	-----	-----	-----	-----

Étapes		Chaque-Blanc	
1	2	1	2
1	B	6666	
2	o	6666	
		+ +	
		11177	
		= =	
3	n	17243	
		17243	
		+ +	
		1101	
		= =	
		28274	
		28274	
		mod	
		25056	
		= =	
4	j	3118	
		3118	
		+ +	
		10637	
		= =	
5	o	13755	
		155	
		+ +	
		248	
		137	
		= =	
		403	
		111	
6	u	403	
		= mod	
		248	
		256	
		=	
		147	
		24847	
		+ +	
		11709	
		= =	
		36256	
		36256	
		mod	
		mod	
		25056	
		= =	
		109	

		100
		+ +
7	r	11223
		= =
		22323

On retrouve bien la somme de contrôle initiale, 223, comme calculé auparavant. La somme de contrôle de Fletcher nous donne aussi 223, ce qui est un hasard. Un protocole qui utilise cet algorithme ajoutera ces valeurs à la fin du message.

Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6	Octet 7	Octet 8	Octet 9
B	o	n	j	o	u	r	223	223

TABLE IX.2.5. – Message tel qu’il est transmis, en incluant la somme de contrôle de Fletcher  
Nous avons évoqué le fait que l’ajout de cette seconde somme de contrôle réduit les risques de collision. En effet, avec une somme de contrôle basique, deux erreurs peuvent se neutraliser et donner quand même le bon résultat, notamment si deux blocs sont inversés. Avec Fletcher, comme la deuxième *checksum* conserve en quelque sorte un historique des opérations, cela devient fort peu probable. Voyons ce qu’il se passe si, dans notre exemple, deux lettres se retrouvent inversées lors de la transmission.

Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6	Octet 7	Octet 8	Octet 9
B	o	j	n	o	u	r	223	223

TABLE IX.2.7. – Le message tel qu’il est reçu, avec une inversion

B	o	j	n	o	u	r
066	111	106	110	111	117	114

		Checksum
		1 2
1	B	6666
		6666
		+ +
2	o	11177
		= =
		17243

3	j	17243
		+ +
		1007
		= =
		28370
		28370
		mod
		mod
		25056
		= =
4	n	2714
		2714
		+ +
		11037
		= =
		13751
		151
		+
		248
		137
5	o	+
		399
		111
		399
		=
		mod
		248
		256
		=
		143
6	u	248
		+
		117
		143
		=
		+
		365
		109
		365
		=
7	r	mod
		252
		256
		=
		109
		252
		+
		223
		109
		=
		+
		475
		114
		475
		=
		mod
		223
		256
		=
		219

Cette erreur de transmission n'a pas été détectée par la somme de contrôle simple, mais est repérée par celle de Fletcher : on trouve comme résultat 219 au lieu de 223. On sait donc que le message n'est pas bon et qu'il ne faut pas le prendre en compte.

La facilité de calcul de cet algorithme et sa relative fiabilité lui ont valu d'être publié en 1982

dans la revue IEEE *Transactions on Communications*, qui est la référence dans le domaine des télécommunications. En 1990, une proposition a été émise pour qu’il puisse être utilisé dans TCP, mais cela n’a pas abouti. En 2018, les systèmes de fichiers ZFS et APFS utilisent toujours l’algorithme de Fletcher. Ce dernier a inspiré d’autres personnes, comme Mark Adler, qui en a proposé sa propre version.

IX.2.3. L’algorithme Adler-32

L’algorithme de Fletcher permet, en théorie, une certaine souplesse : on peut découper les messages en des blocs de la taille qu’on veut (1 octet, 2 octets, ...), on peut choisir sur combien d’octets tiendra la somme de contrôle, etc. En 1995, l’ingénieur américain Mark Adler propose cette configuration particulière :

- Chaque somme de contrôle est calculée sur 2 octets ;
- La *checksum* 1 prend pour valeur de départ 1, tandis que la *checksum* 2 part de 0 ;
- Une opération est réalisée pour chaque bloc de 8 *bits*, soit 1 octet ;
- Le diviseur utilisé pour les modulus est 65521 ;
- Au bout du message à transmettre, la *checksum* 2 est inscrite avant la *checksum* 1.

Reprenons notre exemple et appliquons l’algorithme Adler-32.

B	o	n	j	o	u	r
066	111	110	106	111	117	114

Checksum		Checksum	
Étape		Étape	
1		2	
		1	
		+	
1	B	66	67
		=	
		67	
		6767	
		+	
		+	
2	o	11	17
		=	
		17845	
		17845	
		+	
		+	
3	n	11	08
		=	
		28533	
		28533	
		+	
		+	
4	j	10	09
		=	
		39927	

1. L’ **ASCII** est une norme qui fait correspondre des caractères, comme des lettres, à des nombres, exploitables par les ordinateurs. En **ASCII**, la lettre A correspond au nombre 65, le B à 66, etc.

		39927
		+ +
5	o	11505
		= =
		505.432
		505.432
		+ +
6	u	11622
		= =
		622.054
		622.36
		+ +
7	r	112.054
		= =
		736.790

Le message transmis est alors :

O	O	O	O	O	O	O	O	O	O	O	O	c
t	t	t	t	t	t	t	t	t	t	t	t	t
1	2	3	4	5	6	7	8	9	10	11		
B	o	n	j		o	u	r	2.79	0736			

TABLE IX.2.13. – Message transmis, incluant la somme de contrôle d’Adler-32. La deuxième *checksum* est inscrite avant la première.

Ce paramétrage utilise la valeur 65521 pour les modulus. Dans notre exemple, nous n’atteignons pas cette valeur, donc nous le visualisons pas. Sans rentrer dans les détails, l’utilisation de ce nombre premier réduit le risque de collision. Toutefois, cette configuration rend l’exécution de l’algorithme plus lente que la plupart des paramétrages communs de Fletcher.



Pourquoi on en parle, alors ?

Eh bien voyez-vous, Adler-32 est très utilisé. Mais vraiment **très, très** utilisé. Et pour cause : on le retrouve dans **zlib**, une bibliothèque logicielle qui est utilisée pour la compression de pas mal de choses, comme les archives ZIP ou les images PNG. Ce programme est un composant essentiel de bon nombre de systèmes d'exploitation et de consoles de jeu du 21ème siècle.

Et vous vous demandez, pourquoi Adler-32 se retrouve dans zlib ? C'est très simple : zlib a été créée par 2 ingénieurs, le polytechnicien français Jean-loup Gailly... et un certain Mark Adler. 🍊 Tant qu'à faire, les deux compères ont aussi créé gzip, un format de compression utilisé sur la couche 6 (présentation) du modèle OSI. 🍊

Ces deux algorithmes, Fletcher et Adler, ont pour avantages la simplicité et la rapidité d'exécution, mais ne sont pas totalement fiables. De plus, ils permettent de détecter une erreur dans un message, mais pas de la localiser et encore moins de la corriger. Pour cela, on préférera utiliser un contrôle de redondance cyclique.

IX.2.4. Quand IP rime avec simplicité

! Cette section est liée [au chapitre sur le protocole IP](#) . Il est recommandé de connaître l'en-tête IPv4 avant de poursuivre.

Le protocole IP dans sa version 4 utilise une somme de contrôle, mais pour l'en-tête des paquets uniquement. Le principe est relativement simple : on découpe le *header* par blocs de 16 *bits*, et on les additionne tous. Pour faciliter les choses, on écrira ces calculs en binaire. Si le résultat s'écrit sur plus de 16 *bits*, on découpe et on additionne encore, mais en commençant par la droite. On termine en inversant la valeur de chaque *bit* et on a la somme de contrôle de l'en-tête IP. Faisons un exemple en prenant un paquet IP reproduit dans le tableau ci-dessous. Pour information, il s'agit d'une réponse écho ICMP d'un serveur de Zeste de Savoir.

Off-set								1								2								3								
Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	IP v. 4			IHL 5			DSCP : 0					ECN 0		Longueur totale : 60																	
4	32	Identification : 35802 <sub>10</sub> (8bda <sub>16</sub> )													Flags 0		Fragment offset : 0															
8	64	Time To Live : 52						Protocole ICMP (1)						Header checksum																		
12	96	Adresse IP source : 92.243.7.44																														
16	128	Adresse IP destination : 192.168.1.10																														

TABLE IX.2.15. – En-tête complété d'un paquet IP

Nous n'indiquons pas quelle est la *checksum*, car nous allons la calculer, comme si nous étions à la place du routeur qui a transmis ce paquet. Pour cela, on considère qu'elle vaut zéro.

## IX. Annexes

Maintenant, retranscrivons ces valeurs en binaire et faisons abstraction des champs. Cela nous donne ceci :

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
4	32	1	0	0	0	1	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	64	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	96	0	1	0	1	1	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0	1	0
16	128	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1

On découpe cela par blocs de 16 *bits*, qu'on additionne. Nous allons donc calculer la somme des nombres suivants :

— 0100010100000000  
 — 0000000000111100  
 — 1000101111011010  
 — 0000000000000000  
 — 0011010000000001  
 — 0000000000000000  
 — 0101110011110011  
 — 0000011100101100  
 — 1100000010101000  
 — 0000000100001010

Vous pouvez faire le calcul à la main pour vous entraîner. Sinon, une calculatrice binaire fait l'affaire. 🍊

Cela nous donne **10 0010 1010 1110 1000**. Comme ce résultat s'écrit sur 18 *bits* et qu'il nous en faut 16, on va encore le découper, mais en commençant par la droite. On doit donc additionner les valeurs suivantes :

— 0010 1010 1110 1000  
 — 10

Vous trouverez aisément sans calculatrice que cela fait **0010 1010 1110 1010**. Dernière chose, on inverse la valeur de chaque *bit*. Les 1 deviennent des 0, les 0 deviennent des 1. Cette opération s'appelle **complément à 1**. Nous obtenons ainsi comme valeur :

**1101 0101 0001 0101**

On peut aussi l'écrire en hexadécimal : D515<sub>16</sub>. Et voilà notre somme de contrôle de l'en-tête ! La preuve avec la capture du paquet d'origine, récupérée avec le logiciel Wireshark :

```

▼ Internet Protocol Version 4, Src: 92.243.7.44, Dst: 192.168.1.10
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x8bda (35802)
  > Flags: 0x0000
    Time to live: 52
    Protocol: ICMP (1)
    Header checksum: 0xd515 [validation disabled]
    [Header checksum status: Unverified]
    Source: 92.243.7.44
    Destination: 192.168.1.10

```

FIGURE IX.2.1. – IP header checksum

On peut voir sur cette image qu'IP n'a même pas pris la peine de vérifier cette somme de contrôle. Qu'à cela ne tienne, faisons-le nous-mêmes ! Pour cela, l'opération est exactement la même, on y inclut juste la *checksum* dans notre addition. Si tout se passe bien, le résultat final doit être égal à 0.

Nous avons déjà calculé la somme de tout le reste auparavant, cela donnait 10 0010 1010 1110 1000. Ajoutons-y la somme de contrôle transmise :

```

— 10 0010 1010 1110 1000
— + 1101 0101 0001 0101

```

Cela donne **10 1111 1111 1111 1101**. Pareillement, on découpe 16 *bits* en partant de la droite, ce qui nous mène à la somme suivante :

```

— 1111 1111 1111 1101
— + 10

```

Ce qui fait **1111 1111 1111 1111**. On termine en faisant un complément à 1, ce qui fait **0000 0000 0000 0000**. On tombe bien sur 0, la somme de contrôle valide bien l'intégrité de l'en-tête !



## Conclusion

Nous avons fait le tour de quelques méthodes de computation de somme de contrôle. Désormais, vous avez des notions de mathématiques qui vous serviront dans bien des domaines comme la cryptographie. L'aspect arithmétique des réseaux est peu attrayant et souvent occulté. Il est pourtant important que vous ayez quelques bases, c'est pourquoi cette annexe est là. 🍊

## Contenu masqué

### Contenu masqué n°8

- $13 \bmod 4 = 1$ , car  $13 / 4 = 3$  et il reste 1
- $50 \bmod 10 = 0$ , car  $50 / 10 = 5$  et il reste 0
- $735 \bmod 256 = 223$ , car  $735 / 256 = 2$  et il reste 223

[Retourner au texte.](#)

## IX.3. Analyse fine des communications réseaux

### Introduction

Au long de ce cours, vous avez pu rencontrer des analyses de trames, qui permettent de visualiser concrètement les données qui transitent sur le réseau. Elles ont été réalisées au moyen du logiciel Wireshark. Il serait utile que vous sachiez vous aussi utiliser un tel analyseur : cela vous permettrait de savoir précisément ce qui est émis ou reçu par votre poste.

Dans ce chapitre, nous allons voir deux analyseurs de trames : Wireshark, qui dispose d'une interface graphique très performante, et tcpdump, qui fonctionne en mode texte (en console), ce qui se révèle bien pratique pour des serveurs.

#### IX.3.1. Présentation générale

Le principal intérêt de ces logiciels est de visualiser les trames qui transitent par les interfaces réseaux d'un appareil. Cela est utile pour comprendre et résoudre un dysfonctionnement, mais cela est aussi très instructif pour connaître finement un protocole. On peut aussi s'en servir pour faire de la rétro-ingénierie, c'est-à-dire déterminer le fonctionnement d'une application en observant son comportement.

Les analyseurs permettent de capturer les flux qui passent en temps réel, de les sauvegarder au format *pcap*<sup>1</sup>, de lire des enregistrements, de les filtrer et de les analyser finement. Pour les hôtes ne disposant pas d'une interface graphique, comme souvent les serveurs, nous recommandons **tcpdump**. L'installation se fait au moyen du gestionnaire de paquets. Nous supposons que si vous savez utiliser un serveur, vous savez installer un logiciel. 🍏

Pour les systèmes à interface graphique (Windows, Ubuntu, Mac OS X, ...), nous conseillons [Wireshark](#) . Rendez-vous sur [cette page de téléchargement](#)  pour l'installer.

#### IX.3.2. Utilisation de tcpdump

Que diriez-vous de faire joujou avec notre nouvelle application ? 🍏 Nous allons commencer par présenter tcpdump.

Les exemples présentés ici sont capturés depuis un serveur exposé sur Internet. Pour lancer notre analyse, on lance tout simplement la commande **tcpdump**. Pour l'arrêter, on utilise la combinaison `CTRL` + `C` (ou `⌘` + `C` sous Mac).

```
1 tcpdump: verbose output suppressed, use -v or -vv for full
  protocol decode
```

---

1. pcap est un format de fichier permettant de stocker des trames réseaux.

```

2  listening on ens3, link-type EN10MB (Ethernet), capture size
   262144 bytes
3  12:58:37.199174 IP netmon-1-bhs.ovh.ca > 105.ip-71-57-81.eu: ICMP
   echo request, id 57303, seq 1, length 12
4  12:58:37.199240 IP 105.ip-71-57-81.eu > netmon-1-bhs.ovh.ca: ICMP
   echo reply, id 57303, seq 1, length 12
5  12:58:37.199856 IP 105.ip-71-57-81.eu.44326 > cdns.ovh.net.domain:
   59980+ PTR? 105.71-57-81.in-addr.arpa. (43)
6  12:58:37.201701 IP cdns.ovh.net.domain > 105.ip-71-57-81.eu.44326:
   59980 1/0/0 PTR 105.ip-71-57-81.eu. (75)
7  12:58:37.201887 IP 105.ip-71-57-81.eu.53832 > cdns.ovh.net.domain:
   6713+ PTR? 1.37.114.167.in-addr.arpa. (43)
8  12:58:37.203776 IP cdns.ovh.net.domain > 105.ip-71-57-81.eu.53832:
   6713 1/0/0 PTR netmon-1-bhs.ovh.ca. (76)
9  12:58:37.204270 IP 105.ip-71-57-81.eu.42985 > cdns.ovh.net.domain:
   46110+ PTR? 99.33.186.213.in-addr.arpa. (44)
10 12:58:37.206165 IP cdns.ovh.net.domain > 105.ip-71-57-81.eu.42985:
   46110 1/0/0 PTR cdns.ovh.net. (70)
11 12:58:39.857063 IP netmon-1-gra.ovh.net > 105.ip-71-57-81.eu: ICMP
   echo request, id 49924, seq 1, length 12
12 12:58:39.857156 IP 105.ip-71-57-81.eu > netmon-1-gra.ovh.net: ICMP
   echo reply, id 49924, seq 1, length 12
13 12:58:39.857572 IP 105.ip-71-57-81.eu.52686 > cdns.ovh.net.domain:
   61030+ PTR? 1.186.222.92.in-addr.arpa. (43)
14 12:58:39.859535 IP cdns.ovh.net.domain > 105.ip-71-57-81.eu.52686:
   61030 1/0/0 PTR netmon-1-gra.ovh.net. (77)
15 ^C
16 12 packets captured
17 12 packets received by filter
18 0 packets dropped by kernel

```



Cette capture a été prise complètement au hasard. Pour des raisons de confidentialité, des adresses et des indications horaires ont pu être modifiées.

On peut voir une série de lignes toutes structurées de la même manière :

heure (au millionième de seconde !)	protocole de niveau 3	source
12:58:37.199174	IP	netmon-1-bhs.ovh.ca
12:58:37.199240	IP	105.ip-71-57-81.eu
12:58:37.199856	IP	105.ip-71-57-81.eu.44326
...	...	...

Si on veut plus de détails sur ces paquets, on peut ajouter l'option `-vv` à notre commande :

```

1 # tcpdump -vv
2 tcpdump: listening on ens3, link-type EN10MB (Ethernet), capture
  size 262144 bytes
3 13:00:54.067669 IP (tos 0x8, ttl 5, id 1, offset 0, flags [DF],
  proto ICMP (1), length 32)
4   netmon-1-bhs.ovh.ca > 105.ip-71-57-81.eu: ICMP echo request,
    id 43960, seq 1, length 12
5 13:00:54.067736 IP (tos 0x8, ttl 64, id 10606, offset 0, flags
  [none], proto ICMP (1), length 32)
6   105.ip-71-57-81.eu > netmon-1-bhs.ovh.ca: ICMP echo reply, id
    43960, seq 1, length 12
7 13:00:54.068731 IP (tos 0x0, ttl 64, id 7857, offset 0, flags
  [DF], proto UDP (17), length 71)
8   105.ip-71-57-81.eu.51013 > cdns.ovh.net.domain: [bad udp cksum
    0x3c79 -> 0xb297!] 9513+ PTR? 105.71-57-81.in-addr.arpa.
    (43)
9 13:00:54.070665 IP (tos 0x0, ttl 55, id 52876, offset 0, flags
  [none], proto UDP (17), length 103)
10   cdns.ovh.net.domain > 105.ip-71-57-81.eu.51013: [udp sum ok]
    9513 q: PTR? 105.71-57-81.in-addr.arpa. 1/0/0
    105.71-57-81.in-addr.arpa. PTR 105.ip-71-57-81.eu. (75)
11 13:00:54.070967 IP (tos 0x0, ttl 64, id 7858, offset 0, flags
  [DF], proto UDP (17), length 71)
12   105.ip-71-57-81.eu.32925 > cdns.ovh.net.domain: [bad udp cksum
    0x3c79 -> 0x91b6!] 23524+ PTR? 1.37.114.167.in-addr.arpa.
    (43)
13 13:00:54.073168 IP (tos 0x0, ttl 55, id 40370, offset 0, flags
  [none], proto UDP (17), length 104)
14   cdns.ovh.net.domain > 105.ip-71-57-81.eu.32925: [udp sum ok]
    23524 q: PTR? 1.37.114.167.in-addr.arpa. 1/0/0
    1.37.114.167.in-addr.arpa. PTR netmon-1-bhs.ovh.ca. (76)
15 13:00:54.074404 IP (tos 0x0, ttl 64, id 7859, offset 0, flags
  [DF], proto UDP (17), length 72)
16   105.ip-71-57-81.eu.60036 > cdns.ovh.net.domain: [bad udp cksum
    0x3c7a -> 0x3c95!] 43386+ PTR? 99.33.186.213.in-addr.arpa.
    (44)
17 13:00:54.076506 IP (tos 0x0, ttl 55, id 40371, offset 0, flags
  [none], proto UDP (17), length 98)
18   cdns.ovh.net.domain > 105.ip-71-57-81.eu.60036: [udp sum ok]
    43386 q: PTR? 99.33.186.213.in-addr.arpa. 1/0/0
    99.33.186.213.in-addr.arpa. PTR cdns.ovh.net. (70)
19 ^C
20 8 packets captured
21 8 packets received by filter
22 0 packets dropped by kernel

```

Ainsi, on obtient davantage de détails sur différents protocoles : on voit notamment les champs IP et des erreurs de checksum au niveau d'UDP.

On peut enregistrer une capture dans un fichier avec l'option -w : `tcpdump -w frames.cap`

enregistrera les trames capturées dans le fichier `frames.cap`. Il pourra ensuite être lu avec l'option `-r : tcpdump -r frames.cap`.

Si on souhaite ne conserver que certains flux, on peut appliquer un filtre à la capture. Voici quelques exemples :

- Pour ne conserver que les paquets qui concernent l'hôte `192.168.1.1` : `tcpdump host 192.168.1.1`
- Pour n'avoir que les flux TCP vers ou depuis le port `443` : `tcpdump tcp port 443`
- Pour exclure les flux SSH (très pratique quand on est justement connecté en SSH 🍊) : `tcpdump not tcp port 22`

La référence pour connaître toutes les possibilités et toutes les options reste le manuel (`man tcpdump`). Pour plus d'exemples et plus de précisions sur l'utilisation de `tcpdump`, nous vous recommandons [cet article de Lea Linux](#) ↗.

### IX.3.3. Utilisation de Wireshark

On en parle tout au long du cours, Wireshark est l'outil le plus simple à utiliser pour faire de l'analyse réseau. Ouvrons-le tout de suite et réalisons une écoute sur une interface. Pour cela, double-cliquez sur l'interface à écouter dans la liste proposée.

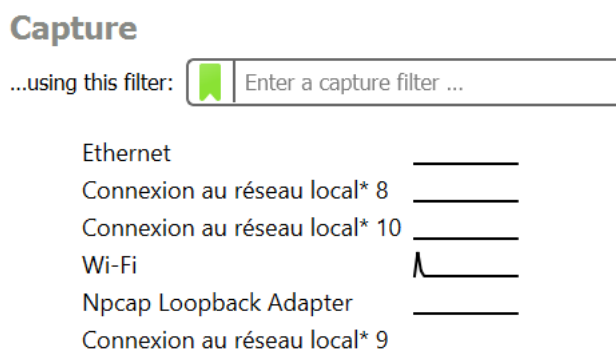


FIGURE IX.3.1. – Liste d'interfaces proposées par Wireshark



S'il manque des interfaces, c'est peut-être que vous n'avez pas les droits suffisants pour exécuter le logiciel. Dans ce cas, essayez de le lancer en mode administrateur (Windows) ou en root (Linux).

Pour arrêter la capture, cliquez sur le carré rouge dans la barre d'outils principale. Juste en dessous, vous pouvez filtrer les paquets à l'aide de mots clés comme **dhcp**, **ip**, **tcp**, ... Pour connaître toutes les possibilités, vous pouvez vous référer à [la documentation officielle](#) ↗. Vous y apprendrez à faire des filtres sur des sous-réseaux (exemple : `ip.addr == 192.168.1.0/24`) ou encore à n'afficher que des segments TCP avec le *flag* SYN levé.

Dans la partie centrale, on visualise le détail des trames couche par couche. La capture suivante est tirée d'un autre chapitre de ce cours.

```
> Frame 174: 1423 bytes on wire (11384 bits), 1423 bytes captured (11384 bits) on interface 0
> Ethernet II, Src: Sagemcom_7[redacted]b (b0:b2:8f:7[redacted]b), Dst: IntelCor_[redacted]a (30:24:32:[redacted]a)
> Internet Protocol Version 6, Src: 2a01:111:2010:8::ff20, Dst: 2a01:cb14:457:ff00:f15d:4f88:b57:8f2b
> Transmission Control Protocol, Src Port: 443, Dst Port: 53016, Seq: 16276, Ack: 5862, Len: 1349
> [8 Reassembled TCP Segments (11429 bytes): #167(1440), #168(1440), #169(1440), #170(1440), #171(1440), #172(1440), #173(1440), #174(1349)]
> Transport Layer Security
```

FIGURE IX.3.2. – Représentation d'une trame dans Wireshark

Si vous réalisez une analyse sur une interface sans-fil, vous allez vite vous rendre compte qu'il y a anguille sous roche avec la couche 2 : Wireshark montre un autre protocole ! Il s'agit généralement d'Ethernet ou SLL (Linux cooked). Cela est dû au fait que Wireshark n'est pas en mesure de lire réellement les trames au niveau 2, alors il fait semblant en proposant une alternative qui permet quand même de visualiser les adresses source et destination de la couche liaison de données. Cela est détaillé dans [la documentation sur le pseudo-protocole SLL](#) .

i

Il est techniquement possible de visualiser des trames Wi-Fi avec Wireshark si l'on écoute une interface en mode *monitor*. Cela est trop complexe pour être expliqué ici, mais vous pouvez vous référer à [cette page \(en anglais\)](#) .

Vous pouvez afficher le détail d'une couche en double-cliquant dessus, et faire de même pour chaque section qui débute par un symbole >.

```

▼ Internet Protocol Version 6, Src: 2a01:111:2010:8::ff20, Dst: 2a01:cb14:457:ff00:f15d:4f88:b57:8f2b
    0110 .... = Version: 6
    ▼ .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
        .... 0000 00.. .... = Differentiated Services Codepoint: Default (0)
        .... ..00 .... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
        .... 0000 0000 0000 0000 0000 = Flow Label: 0x000000
    Payload Length: 1369
    Next Header: TCP (6)
    Hop Limit: 111
    Source: 2a01:111:2010:8::ff20
    Destination: 2a01:cb14:457:ff00:f15d:4f88:b57:8f2b

```

FIGURE IX.3.3. – Détail du protocole de niveau 3

Tout en bas, vous trouvez une représentation en hexadécimal de la section sélectionnée. Dans l'image suivante, c'est la section IP qui est surlignée en bleu. On y visualise le numéro de version dans le premier quartet (4), l'IHL (5, soit 20 octets), etc. Ne cherchez pas de lien avec la capture précédente, il s'agit d'une trame différente.

0000	b0 b2 8f 7	a 30 24 32	a 08 00 45 00	...s.Z0\$ 2_...E.
0010	00 67 6a 34 40 00 80 06	00 00 c0 a8 01 0c b9 19		.gj4@... .....
0020	b6 4c 57 94 69 8c ab 8e	e9 24 0e d0 8b b7 50 18		.LW.i...\$....P.
0030	01 fc 31 74 00 00 17 03	03 00 3a 7e bf 34 bc b5		.1t... ..~4..
0040	1f 31 48 76 12 d3 3f 13	7f b6 69 8b bb cb fe b5		.1Hv..?..i.....
0050	0a f7 61 fa f9 37 75 6b	db 03 a1 d2 fb c6 7c 28		..a.7uk .....  (
0060	d5 fb 09 e0 9e e0 77 d6	a6 aa 09 ab 59 d4 f9 49		.....w. ....Y..I
0070	33 e4 89 fc 5d			3...]

FIGURE IX.3.4. – Représentation en hexadécimal d'une trame

## Conclusion

Nous ne saurions trop vous conseiller de jouer avec les analyseurs pour comprendre de manière précise ce qui se trame sur votre réseau. N'hésitez pas à explorer par vous-mêmes toutes les possibilités de Wireshark : il propose des fonctionnalités incroyablement pointues de traçage de

## *IX. Annexes*

flux, de statistiques, de reconstitution d'appels téléphoniques, etc. Quand vous découvrez une notion dans ce cours, visualiser sa place au sein d'une communication que vous déclenchez est un excellent moyen de la comprendre et de s'en souvenir.

## IX.4. Qui gouverne Internet ?

### Introduction

Internet est un réseau mondial et décentralisé. Quand on cherche à savoir qui le contrôle, le gouverne ou a autorité dessus, c'est très difficile de trouver une réponse tellement il y a d'acteurs différents. Pour y voir plus clair, nous allons suivre trois axes complémentaires : l'adressage, les standards, et la politique. Dans un premier temps, nous allons voir comment sont attribuées les adresses IP et gérés les noms de domaine. Ensuite, nous nous demanderons qui définit les protocoles et s'il existe une obligation de les respecter. Enfin, nous nous interrogerons sur l'aspect légal dans un réseau qui ne connaît pas de frontières.

#### IX.4.1. Des chiffres et des lettres

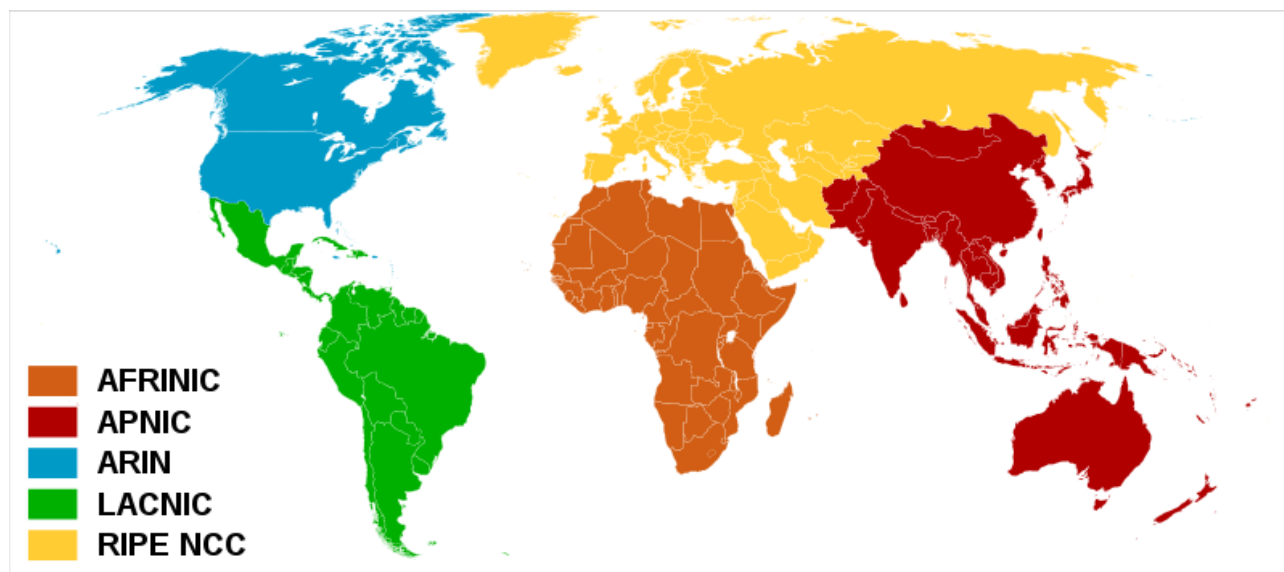
Commençons par une problématique plutôt simple : l'adressage. Comment se fait-il que vous ayez une adresse IP et pas une autre ? Il y a un organisme en charge de l'attribution des adresses : l' **ICANN**.

Il s'agit d'une société à but non lucratif, c'est-à-dire qu'elle ne cherche pas à gagner de l'argent. Elle est basée aux États-Unis, et plus précisément à Los Angeles, en Californie. Un de ses rôles principaux est de coordonner la distribution des adresses IP. Rassurez-vous, ce n'est pas elle toute seule qui affecte une à une les milliards d'adresses à tour de bras ! 🍊 Elle affecte des blocs d'adresses à des **registres Internet régionaux** ( **RIR**, en anglais *Regional Internet Registry*), qui sont au nombre de 5.

Chacun de ces 5 **RIR** est affecté à une région du monde :

- le **RIPE NCC** a autorité en Europe, au Proche et Moyen-Orient ainsi qu'en Russie ;
- l' **APNIC**, qui officie dans le reste de l'Asie ainsi qu'en Océanie ;
- l' **AfriNIC**, pour l'Afrique ;
- l' **ARIN** est en charge du Canada, des États-Unis ainsi que de certaines îles proches ;
- le **LACNIC** s'occupe du reste de l'Amérique.

L'image suivante [↗](#), issue de l'Atelier Graphique de Wikipédia et distribuée sous [licence CC BY SA 3.0](#) [↗](#), représente ces différentes plaques géographiques.



Ces organismes reçoivent une délégation de l' **ICANN** pour l'attribution de différents blocs d'adresses IP ainsi que des numéros de systèmes autonomes, utilisés pour le routage au niveau mondial. Là encore, ils ne gèrent pas tout eux mêmes. Ils délèguent des blocs d'adresses à d'autres entités : les **LIR** (*Local Internet Registry*, registres Internet locaux). Ces entités correspondent généralement aux opérateurs de télécommunications, qui attribuent des adresses IP à leurs clients finaux.



Dans certains pays, il existe un intermédiaire entre les **RIR** et les **LIR** : ce sont les **registres Internet nationaux** (en anglais *National Internet Registry*, **NIR**). C'est le cas notamment en Asie (Vietnam, Japon, Chine, ...) et en Amérique latine (Brésil, Mexique, ...).

En dehors des adresses IP, l' **ICANN** a aussi pour rôle la gestion et la coordination des noms de domaines. En pratique, là encore, elle délègue à différents organismes la distribution en fonction du domaine de premier niveau. Ainsi, la société états-unienne Verisign a la charge notamment du .com, du .net et du .tv. Les domaines français que sont .fr, .re, .mf, etc.<sup>1</sup>, sont gérés par l' **Afnic**. De manière générale, chaque pays dispose d'une autorité pour gérer ses domaines de premier niveau : l' **ANRT** marocaine s'occupe du .ma, DNS Belgium, du .be, NIC Sénégal est en charge du .sn, etc. Les domaines dits "génériques" (.info, .biz, ...) sont gérés par des sociétés privées.



Les domaines à deux lettres sont exclusivement réservés aux pays. On parle alors de **ccTLD**, pour Country Code Level Top Domain. Vous croyiez que le .tv faisait référence à la télévision ? Perdu, il s'agit de l'État des Tuvalu ! Ce petit pays d'Océanie risque de disparaître à cause de la montée des eaux, ce qui pourrait avoir pour conséquence... la disparition du domaine .tv ! 🍌 Un tel phénomène peut arriver sans crier gare : en octobre 2024, le Royaume-Uni a cédé à la république de Maurice (un archipel de l'océan Indien) un territoire disputé de longue date. Problème : le domaine .io était affecté à ce territoire, dont le nouveau statut ne permet pas de conserver son propre ccTLD ! Cet



accord politique a, possiblement sans qu'aucun responsable ne s'en rende compte, creusé la tombe du domaine .io. Plus d'infos dans ce billet [↗](#).

Ces différentes entités permettent, moyennant une redevance, à d'autres entreprises ou associations de vendre à des particuliers et des professionnels des noms de domaine, tels que [zestedesavoir.com](#). Ces revendeurs sont appelés **registraires** ou *registrars* en anglais. Parmi ces sociétés, on peut citer OVH ou Gandi.

## IX.4.2. Des standards et des normes

Gouverner le réseau, c'est aussi définir des normes et des standards. En ce qui concerne les protocoles utilisés sur Internet pour permettre à tout le monde de communiquer, c'est le rôle de l' **IETF**. Cet organisme de normalisation est basé aux États-Unis mais il est ouvert : n'importe qui peut y participer et contribuer à ses travaux. Ce groupe de travail est rattaché à l'Internet Society (abrégiée **ISOC**), un organisme à but non lucratif créé par Vint Cerf et Bob Kahn, deux des pères fondateurs d'Internet. Ils ont quand même inventé TCP/IP, ce n'est pas rien. 🍊 L' **ISOC**, et l' **IETF** par là-même, fait autorité *de facto* dans le monde du réseau.

L' **IETF** édite des **RFC**, des documents de référence qui définissent la manière dont doivent fonctionner des réseaux ou des protocoles. Il appartient ensuite aux industriels d'implémenter ou non ces **standards**. Personne n'a d'obligation légale de suivre à la lettre une **RFC**, mais si un constructeur de matériel ou un éditeur de logiciel veut que son produit fonctionne normalement avec d'autres solutions, il a intérêt à la respecter ! 🍊

Comme tout le monde peut participer aux travaux de l' **IETF**, il est tout à fait possible de proposer une amélioration à une **RFC** existante. Dans ce cas, une nouvelle **RFC** est éditée, en précisant laquelle ou lesquelles elle rend obsolètes. Là encore, elle deviendra un standard si elle est adoptée et implémentée globalement. Il en existe pour tout et n'importe quoi : cela va de la définition exacte du protocole IP au vocabulaire à employer pour écrire des **RFC** (numéro 2119), en passant par des poissons d'avril comme le transport de paquets par pigeon voyageur ([véridique ! ↗](#)). Et ça fait plus de 50 ans que ça dure : la **RFC** 1 date d'avril 1969 ! L'intégralité de ces documents est publiée [sur le site web de l' IETF ↗](#).

Si on se penche vers l'aspect physique des transmissions réseaux, on doit s'intéresser à un autre institut : l' **IEEE**. C'est une organisation basée aux États-Unis mais qui est plus ancienne que les autres, car son domaine d'autorité est l'électronique, et non l'informatique. Ainsi, son comité numéro 1394 est en charge du FireWire, un protocole de transfert de données à vitesse importante et constante, utilisé notamment par des caméras. Elle est tout de même incontournable dans le monde du réseau pour son comité 802. C'est lui qui normalise tout ce qui touche aux réseaux locaux (LAN) au niveau électronique. Dans ce cours, on a fait référence à lui à plusieurs reprises : c'est de là que viennent les normes 802.11 (Wi-Fi), le 802.1q ("dot1q" quand on parle de VLAN), ou encore 802.3 (Ethernet).

---

1. Le .fr est le domaine générique français, mais il en existe de nombreux autres qui correspondent à des départements ou collectivités d'outre-mer : .gf (Guyane), .gp (Guadeloupe), .mq (Martinique), .pm (Saint-Pierre-et-Miquelon), .re (Réunion), .wf (Wallis-et-Futuna), .yt (Mayotte), .mf (Saint-Martin), .bl (Saint-Barthélemy) et .tf (Terres Australes et les Antarciques Françaises).



Il y a une différence entre une **norme** et un **standard** ! En anglais, ces deux notions sont désignées par le terme générique ”*standard*”, mais en français, on est plus précis. La norme a une dimension officielle, elle doit obligatoirement être respectée. Quand l’ **IEEE** édite une norme relative à la puissance d’émission d’une carte Wi-Fi, par exemple, un constructeur doit impérativement la suivre à la lettre. Sans cela, non seulement son produit risque de ne pas fonctionner correctement, mais en plus, cela pourrait être dangereux pour la santé et la sécurité des personnes. *A contrario*, si un industriel n’implémente pas un standard de l’ **IETF**, cela ne présente pas de danger pour les utilisateurs. Au pire, son produit ne fonctionnera pas bien et les gens ne l’utiliseront pas. Tant pis pour lui ! 🍊

Un petit schéma pour récapituler tout ce que nous avons vu jusqu’à présent ? 🍊

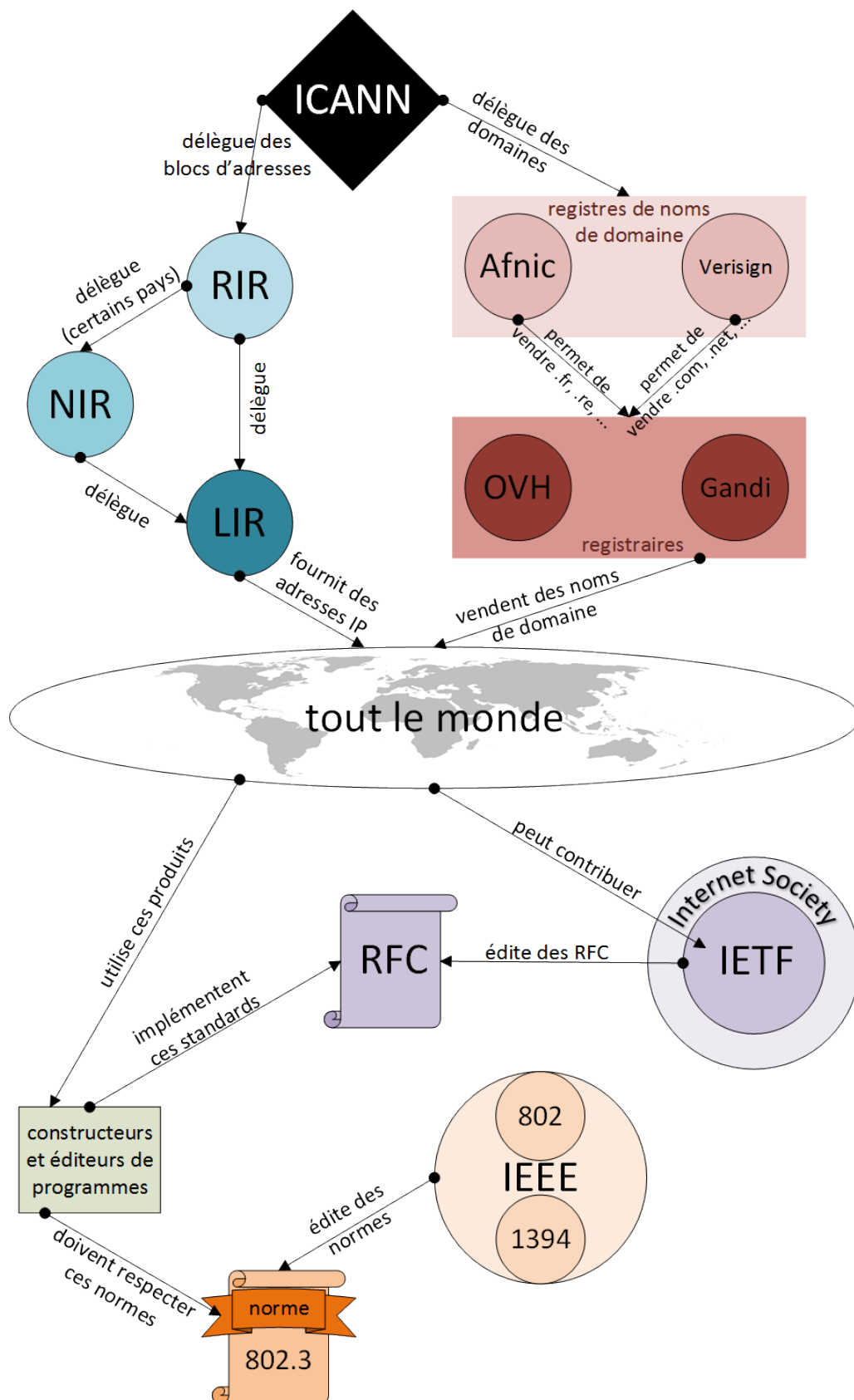


FIGURE IX.4.1. – Relations entre les principaux acteurs d'Internet

### IX.4.3. Des politiques et des lois

Nous avons vu qui décidait de l'adressage, des protocoles et des normes, mais qu'en est-il du contrôle des contenus ? Cela est très variable et relève de la souveraineté de chaque pays.

Internet a été conçu pour transporter des données de manière neutre. Rien ne permet *a priori* de discriminer des flux en fonction de leur contenu. Personne ne peut "contrôler" Internet tout entier, mais les États peuvent théoriquement décider de ce qui est autorisé ou non sur leur territoire. Ainsi, la Chine a très tôt instauré un système de contrôle très sévère connu sous le nom de "grande muraille" (*Great Firewall of China*). Les ressources accessibles depuis son territoire sont très restreintes.

D'autres pays pratiquent la censure de manière moins sévère. En France, les fournisseurs d'accès à Internet peuvent recevoir une injonction d'empêcher l'accès à un site web. En pratique, ils se contentent de modifier ou retirer le nom de domaine de leurs serveurs DNS. Ainsi, l'utilisation d'un autre serveur, comme OpenDNS ou CloudFlare permet de contourner aisément cette restriction.

De nos jours, la plupart du trafic transitant sur le réseau est chiffré. Il n'est pas possible de décrypter à la volée tout ce qui passe pour en bloquer une partie. Les agissements illicites sont détectés *a posteriori*. En cas de publication contraire à la loi, par exemple des propos racistes sur un réseau social, la législation peut prévoir une obligation de retrait par la personne qui propose le service (hébergeur, responsable de publication, ...). Toutefois, cette contrainte ne peut s'appliquer que sur un territoire donné : si un individu publie depuis une connexion en France un contenu illégal selon la loi française, mais sur une plateforme aux États-Unis, cette dernière n'a aucun ordre à recevoir d'un pays étranger.

Il n'est donc théoriquement pas possible de contrôler totalement les flux qui transitent par le réseau. Cela ne peut normalement se faire qu'à l'échelle d'un État souverain. D'un point de vue strictement technique, tout blocage peut être contourné, de manière plus ou moins simple, dès qu'on peut joindre un serveur en dehors de la zone d'influence du blocage en question.

## Conclusion

Ce chapitre est loin d'être exhaustif, les entités qui ont une influence sur Internet sont trop nombreuses pour toutes les citer. Nous avons évoqué les organismes principaux qui font autorité sur le réseau, au travers de la gestion de l'adressage, des protocoles, des standards et des normes. La problématique de la législation sur un réseau mondial est complexe, nous avons tenté de vous donner des pistes générales pour comprendre comment elle peut possiblement s'appliquer. Nous espérons que vous avez une idée plus claire de qui "gouverne" Internet. 🍊

## Conclusion

## Conclusion

Ce cours a été initialement rédigé pour le feu Site du Zéro, devenu OpenClassrooms. Ce dernier ayant fait n'importe quoi au point de rendre le contenu inintelligible, il a été décidé de le porter sur Zeste de Savoir. La rédaction est maintenant terminée. Les contenus publiés sont maintenus dans la mesure du possible grâce à vos retours. Vous pouvez vous tenir au courant des nouvelles publications sur [notre page Facebook](#) et [notre compte Twitter](#).

Suite au succès de ce cours, notamment son utilisation comme référence dans des entreprises et universités, **Les Réseaux de Zéro** sont aussi édités en livre aux éditions Eyrolles depuis le 17 février 2022. Cette édition reprend en grande partie le cours existant en ligne, avec quelques adaptations, et inclut en plus des exercices pratiques. Sa déclinaison axée hacking, **La Cybersécurité de Zéro**, est disponible depuis le 2 mai 2024. [Plus d'infos ici !](#)

---

Nous, Junior et Vince, auteurs, tenons à remercier toutes les personnes qui sont intervenues sur ce tuto depuis sa création, le 14 février 2009. Citons en particulier **The frog**, qui a collaboré avec nous durant 3 ans, [kankan](#) qui est à l'origine du titre de ce tuto, @Taurre qui a eu la lourde tâche d'assurer la validation éditoriale pendant 9 mois, ainsi que Titouan Soulard qui a créé de nombreux schémas. Nous remercions également les anciens validateurs de la période où ce cours était publié sur OpenClassrooms : [Guizmo2000](#), [Zopieux](#), [Coyote](#), [Petrus6](#), [Arthurus](#), [Natalya](#), [Thunderseb](#), [coma94](#), [SimSonic](#), [Oneill887](#) et [Alexis211](#), ainsi que toute l'équipe menée par [Stylla](#) qui a entrepris une correction complète du cours en 2012 - ça a pris 48 jours !

Pour [la version livre du cours](#), nous remercions toute l'équipe éditoriale de la maison d'édition Eyrolles, qui a cru en notre projet et nous a permis de réaliser un rêve. Merci en particulier à Antoine Derouin, responsable d'édition, qui a chapeauté le tout.

Enfin, nous vous remercions, vous, lecteurs, lectrices, pour vos retours, votre soutien, votre fidélité et vos témoignages. Depuis 15 ans, nous lisons l'intégralité de vos messages, qui nous poussent à vouloir vous proposer le meilleur contenu possible. C'est grâce à vous que la rédaction de ce cours a repris en 2018, après 5 longues années d'abandon.

# Liste des abréviations

Afnic	Association française pour le nommage Internet en coopération.	439
AfriNIC	African Network Information Centre.	438
ANRT	Agence Nationale de Réglementation des Télécommunications.	439
APNIC	Asia-Pacific Network Information Center.	438
ARIN	American Registry for Internet Numbers.	438
ASCII	American Standard Code for Information Interchange.	422, 427
ATM	Asynchronous Transfer Mode.	294
BAS	Broadband Access Server.	234
CAT5	Category 5.	23
CAT5E	Category 5E.	23
CRC	Contrôle de Redondance Cyclique / Cyclic Redundancy Check.	47
EBGP	Exterior Border Gateway Protocol.	260
EGP	Exterior Gateway Protocol.	260, 261
FaaS	Function as a Service.	365
FAI	Fournisseur d'Accès à Internet.	260
FQDN	Fully Qualified Domain Name.	325
Gb/s	Gigabits par seconde.	23, 26
IaaS	Infrastructure as a Service.	364
IANA	Internet Assigned Numbers Authority.	168, 170, 271
IBGP	Interior Border Gateway Protocol.	260
ICANN	Internet Corporation for Assigned Names and Numbers.	438, 439
IEEE	Institute of Electrical and Electronics Engineers.	440, 441
IETF	Internet Engineering Task Force.	440, 441
IGP	Interior Gateway Protocol.	235, 239
IP	Internet Protocol.	46, 48, 168, 169, 171, 172, 176, 178
ISN	Initial Sequence Number.	188
ISOC	Internet Society.	440
LACNIC	Latin America and Caribbean Network Information Centre.	438
LIR	Local Internet Registry.	439

## Liste des abréviations

- MAC Media Access Control. 46, 168
- Mb/s Mégabits par seconde. 23
- NDP Neighbor Discovery Protocol. 334
- NIR National Internet Registry. 439
- NTLM NT LAN Manager. 349, 351
- PaaS Platform as a Service. 364
- PAT Port Address Translation. 5, 171
- RFC Request For Comments. 440
- RIPE NCC Réseau IP Européen - Network Coordination Centre. 438
- RIR Registre Internet Régional / Regional Internet Registry. 438, 439
- RJ11 Registered Jack 11. 24
- RJ45 Registered Jack 45. 24
- RTO Retransmit TimeOut. 188
- SaaS Software as a Service. 364
- SDN Software Defined Networking. 362
- SDU Service Data Unit. 64–66
- SMTP Simple Mail Transfer Protocol. 168–170, 176
- SSH Secure Shell. 345–351
- TCP Transmission Control Protocol. 48, 168–172, 174, 176, 178
- TLS Transport Layer Security. 345
- UDP User Datagram Protocol. 169–172, 174, 176