Annexes

Cette partie contient des annexes traitant de notions utilisées en réseau, sans en être fondamentalement. Elles peuvent être consultées indépendamment du reste du livre.

Annexe 1

Binaire et hexadécimal : partez sur de bonnes bases !

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 conclurons avec un point sur le stockage des nombres par les ordinateurs.

Décimal vs binaire : un peu de pratique

Comme nous sommes dans un livre sur les réseaux, 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.

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, que vous dire d'intéressant à son sujet ? Elle est bien pratique pour compter sur ses doigts et puis tout tourne autour d'elle dans le monde, notamment les normes internationales. ☺ En revanche, pour l'informatique, ce n'est pas du tout adapté. Comment, dans un système basé sur les transmissions électriques, est-on censé représenter 10 valeurs différentes ?

Système binaire

Le système binaire est lui aussi un système de numération, sauf qu'il 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 *BInary* et *digiT*). 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 de 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 cet ouvrage ne sont qu'une suite de 0 et de 1. Et nous allons dans ce chapitre apprendre à faire la conversion d'une adresse IP du binaire au décimal et vice versa.

Le système binaire est assez complexe quand il s'agit de représenter des chiffres négatifs. Nous vous invitons à effectuer des recherches sur le sujet à l'occasion. Comme on n'en a pas besoin en réseau, on vous épargne cela. ☺

C'est parti, touchons du binaire

Bon, avant tout, nous allons faire un petit rappel. Une adresse IP est une suite de quatre groupes de chiffres séparés par un point. Chaque « groupe » vaut 1 octet, soit 8 bits. Ce qui nous intéresse ici est le bit.

Un bit ne peut donc prendre que l'une des deux valeurs 0 et 1. Comme nous avons 8 bits, nous allons donc représenter une adresse IP par 32 chiffres (4 groupes de 8 bits) qui ne seront que des 0 et des 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 et l'écrire 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 :

00000000.00000000.00000000.00000000

Voilà, nous y sommes presque. Nous avons écrit 4 groupes de 8 chiffres, donc 32 chiffres. Chacun de ces 32 chiffres ne doit valoir que 1 ou 0. Ici, nous n'avons qu'une suite de zéros, ce qui veut dire

que la 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.

Point vocabulaire important : si un bit vaut 1, on dit qu'il est allumé, sinon c'est qu'il est… éteint, bingo !

Ainsi, pour passer de l'écriture binaire à la forme décimale d'un nombre, 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, 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 précédemment : 12 = 8 + 4. Il nous faut donc allumer le bit dont la correspondance vaut 8 et celui dont la correspondance vaut 4, soit le 3e et le 4e en allant de la droite vers la gauche. Cela nous donne : 0 0 0 0 1 1 0 0. Et voilà ! Procédez de la même façon pour les trois autres octets (ce n'est vraiment pas difficile).

Ceci nous donne au final :

- 1er octet : 00001100 ;
- 2e octet : 00000011 ;
- 3e octet : 00000010 ;
- 4e octet : 00000001.

Donc, en binaire, l'adresse 12.3.2.1 s'écrit : 00001100.00000011.00000010.00000001.

Connaître les puissances de deux vous facilitera vraiment la conversion. Avec un peu de pratique, ça deviendra automatique. Le tableau suivant récapitule la correspondance des bits sur un octet.

Valeur décimale	128	64	32	16	8	4	2	1
Puissance de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Exemple : 87_{10}	0	1	0	1	0	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 précédent aux puissances supérieures.

Valeur décimale	1024	512	256	128	64	32	16	8	4	2	1
Puissance 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 1 337, il suffit de le décomposer en puissances de deux :

```
1 337 = 1024 + 256 + 32 + 16 + 8 + 1
1 337 = 2^10 + 2^8 + 2^5 + 2^4 + 2^3 + 2^0
```

On peut donc écrire $1\ 337_{10} = 101001110012$.

La notation NOMBRE$_{(BASE)}$ 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, $10_{(10)}$ est différent de $10_{(2)}$ (binaire 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 vous habituer. Commencez par écrire un tableau comme les précédents. 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 dans la case 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
Puissance de 2	2^5	2^4	2^3	2^2	2^1	2^0
Nombre au départ : 30_{10}	?	?	?	?	?	?
30 n'est pas plus grand que 32, on continue	0	?	?	?	?	?
30 est supérieur à 16, on calcule 30 - 16 = 14	0	1	?	?	?	?
14 est supérieur à 8, on calcule 14 - 8 = 6	0	1	1	?	?	?

6 **est supérieur à** 4, **on calcule** 6 - 4 = 2	0	1	1	1	?	?
2 **est égal à** 2, **on calcule** 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 $30_{10} = 011110_2$.

On a ici écrit la valeur sur 6 bits, mais on aurait aussi pu le faire sur 5 bits ou plus. Ajouter 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 ajoutant des zéros inutiles. Cela ne change rien aux valeurs : $011110_2 = 11110_2 = 00011110_2$.

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 ajoute un ou plusieurs chiffre(s). En décimal, c'est pareil : on écrit plusieurs chiffres pour représenter un nombre plus grand que 9. En hexadécimal, c'est la même chose, mais avec 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	Hexadécimal
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 et 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.

Hexadécimal	Binaire	Hexadécimal	Binaire
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	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 $FE80_{(16)}$. Qu'est-ce que ça donne en binaire ?

Il suffit de lire le tableau et de remplacer chaque chiffre hexadécimal par sa valeur décimale :

$FE80_{(16)} = 1111\ 1110\ 1000\ 0000_{(2)}$

Comme un chiffre hexadécimal correspond à quatre chiffres binaires, on sépare généralement les blocs de 4 bits pour y voir plus clair. Et voilà !

Avant de clore cette annexe, sachez que les ordinateurs ne peuvent traiter des valeurs numériques que sous forme d'octets. Le nombre 98_{10} peut ainsi être stocké sur 1 octet, car il faut 7 bits pour le représenter. En revanche, le nombre 256_{10} 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 l'annexe suivante.

Annexe 2

Détection d'erreurs avec la somme de contrôle

Nous avons introduit le principe de la somme de contrôle dans le chapitre 12. 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 !*

La vérification de parité

Bienvenue à votre premier baptême du feu. Comme l'indique le titre, nous allons commencer par étudier l'**algorithme de vérification longitudinale de parité**.

La parité, c'est quand il y a autant d'hommes que de femmes, non ?

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 2.

Un algorithme de vérification longitudinale de parité se base sur la parité, mais pas celle des nombres entiers ; il se base sur la **parité de bits**.

Des bits ? Des 0 et des 1 pairs ?

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`. Quatre bits sont allumés (`4` bits qui valent `1`). 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. En revanche, pour un ordinateur qui fonctionne avec des `0` et des `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.

Ça ne nous dit pas ce qu'est un bit de parité…

Il s'agit d'un bit que l'on ajoute au début d'une suite de bits pour nous informer de la parité du nombre de bits allumés. En toute logique, il y en a deux types : le bit de parité pair et le bit de parité impair.

Bit de parité pair

Principe

Quand vous utilisez un bit de parité pair, il aura pour valeur `1` si le nombre de bits allumés est impair.

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`, car `7` bits allumés + le bit de parité pair = `8` bits à `1` et `8` est un nombre pair.

Exercices

Commençons par quelque chose de très simple : `1110000`. Voilà une suite où `3` bits sont allumés. Nous voulons utiliser un bit de parité pair. Nous savons qu'il vaut `1` si et seulement si le nombre de bits allumés est impair, ce qui est le cas ici. Donc, nous allons réécrire cette suite de bits en intégrant le bit de parité pair au début :

```
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. Là encore, 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. Le bit de parité ne peut donc 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.

Bit de parité impair

Principe

Si vous utilisez un bit de parité impair, le bit doit valoir `1` si l'ensemble des bits allumés est pair.

Un exemple ? La suite `1101100` 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.

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 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.

Exercices

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`, qui a `2` bits allumés. `2` est un nombre pair, donc le bit de parité impair vaudra `1` :

```
11100000
```

Nous avons désormais `3` bits allumés, donc un nombre impair. Cela confirme bien la règle : l'ajout d'un bit de parité impair rendra impair le nombre de bits allumés.

Continuons ! Soit la suite `1010100`, avec `3` bits allumés. `3` étant impair, le bit de parité vaudra `0` :

```
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, cela devrait couler de source :

```
11111110
```

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 quatre règles d'or à retenir.

- Un bit de parité pair vaut `1` lorsque le nombre de bits allumés dans une suite est 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 pair. Sinon, il vaut `0`.
- L'ajout d'un bit de parité impair à une suite donnera un nombre impair de bits allumés.

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. Il vaut mieux trouver une information plus fiable. Par ailleurs, on ne veut pas non plus recourir à des méthodes coûteuses en ressources et en temps.

Jusqu'au début des années 1980, un système de somme de contrôle assez simple était utilisé. Il consistait à découper le message à transmettre en blocs de taille fixe (`1` octet, `2` octets, `4` octets…). On calculait ensuite la somme des valeurs de chacun de ces blocs.

Prenons pour exemple le mot `Bonjour`, encodé en ASCII. 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

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.

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 deux !

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 `65 535`. Alors comment faire ?

C'est là qu'intervient l'**arithmétique modulaire**. Ce concept mathématique introduit la fonction modulo. Le **modulo** 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 les opérations suivantes de tête pour vous entraîner :

- `13 mod 4 =` ?
- `50 mod 10 =` ?
- `735 mod 256 =` ?

Voici les réponses :

- `13 mod 4 = 1`, car `13 / 4` donne `3` et il reste `1` ;
- `50 mod 10 = 0`, car `50 / 10` donne `5` et il reste `0` ;
- `735 mod 256 = 223`, car `735 / 256` donne `2` et il reste `223`.

Quel est le 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. Imaginons qu'il y ait une erreur de transmission, par exemple sur le dernier bit de la quatrième lettre du message : le `j` devient un `k` et la valeur numérique décimale est `107` et non `106`. La somme des valeurs est `736` au lieu de `735`. L'opération modulo donne `736 mod 256 = 224` et non `223`. Le récepteur calcule donc 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) inversés.

Nous avons choisi le diviseur `256` dans notre exemple. Cela assure que la somme de contrôle rentrera sur un octet. En effet, tout nombre modulo `256` donne un résultat inférieur ou égal à `255`, qui est la valeur maximale stockable sur un octet. On peut aussi choisir un diviseur inférieur, cela fonctionnerait de la même manière. En revanche, les risques de collision seraient supérieurs, puisque le nombre total de valeurs possibles serait réduit. Si nous avions choisi de stocker notre somme de contrôle sur deux octets, nous aurions pu opter pour un diviseur allant jusqu'à `65 536`.

Vers la fin des années 1970, le physicien John Fletcher, du laboratoire national de Lawrence Livermore en Californie, propose une amélioration à ce système. Elle consiste à ajouter, en plus de la somme de contrôle étudiée précédemment, la somme de toutes les étapes intermédiaires. Ainsi, cela prévient l'inversion de blocs lors de la transmission et réduit le risque de collision. Comme c'est difficile à visualiser, étudions 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 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).

Étape	Bloc	Valeur ASCII	Somme 1	Somme 2
1	B	066	66	66
2	o	111	66 + 111 = 177	66 + 177 = 243
3	n	110	177 + 110 = 287 287 mod 256 = 31	243 + 31 = 274 274 mod 256 = 18
4	j	106	31 + 106 = 137	18 + 137 = 155
5	o	111	137 + 111 = 248	155 + 248 = 403 403 mod 256 = 147
6	u	117	248 + 117 = 365 365 mod 256 = 109	147 + 109 = 256 256 mod 256 = 0
7	r	114	109 + 114 = 223	0 + 223 = 223

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.

Message tel qu'il est transmis, en incluant la somme de contrôle de Fletcher

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

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 qui se passe si, dans notre exemple, deux lettres se retrouvent inversées lors de la transmission.

Étape	Bloc	Valeur ASCII	Somme 1	Somme 2
1	B	066	66	66
2	o	111	66 + 111 = 177	66 + 177 = 243
3	j	106	177 + 106 = 283 283 mod 256 = 27	243 + 27 = 270 270 mod 256 = 14
4	n	110	27 + 110 = 137	14 + 137 = 151
5	o	111	137 + 111 = 248	151 + 248 = 399 399 mod 256 = 143
6	u	117	248 + 117 = 365 365 mod 256 = 109	143 + 109 = 252
7	r	114	109 + 114 = 223	252 + 223 = 475 475 mod 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.

L'algorithme Adler-32

L'algorithme de Fletcher donne, en théorie, une certaine souplesse : on peut découper les messages en des blocs de la taille qu'on veut (1 octet, 2 octets…) et choisir sur combien d'octets tiendra la somme de contrôle. En 1995, l'ingénieur américain Mark Adler propose la configuration particulière suivante :

- 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 le modulo est `65 521` ;
- au bout du message à transmettre, la checksum `2` est inscrite avant la checksum `1`.

Reprenons notre exemple et appliquons l'algorithme Adler-32.

Étape	Bloc	Valeur ASCII	Checksum 1	Checksum 2
1	B	066	1 + 66 = 67	0 + 67
2	o	111	67 + 111 = 178	67 + 178 = 245
3	n	110	178 + 110 = 288	245 + 288 = 533
4	j	106	288 + 106 = 394	533 + 394 = 927
5	o	111	394 + 111 = 505	927 + 505 = 1 432
6	u	117	505 + 117 = 622	1 432 + 622 = 2 054
7	r	114	622 + 114 = 736	736 + 2 054 = 2 790

Le message transmis est alors :

**Message transmis, incluant la somme de contrôle d'Adler-32.
La deuxième somme est inscrite avant la première.**

Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6	Octet 7	Octet 8	Octet 9	Octet 10	Octet 11
B	o	n	j	o	u	r	2 790		736	

Ce paramétrage utilise la valeur `65 521` pour le modulo. Dans notre exemple, nous n'atteignons pas cette valeur, donc nous ne la 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 en parler, alors ?

Eh bien voyez-vous, Adler-32 est très utilisé, vraiment très, très utilisé. Et pour cause : on le retrouve dans **zlib**, une bibliothèque logicielle utilisée pour la compression de beaucoup 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 XXIe siècle.

Vous vous demandez pourquoi Adler-32 se retrouve dans zlib ? C'est très simple : la bibliothèque zlib a été créée par deux ingénieurs, le polytechnicien français Jean-Loup Gailly et… un certain Mark Adler. ☺ 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. Ce dernier ne sera pas développé ici, nous allons plutôt nous pencher sur le système de contrôle utilisé par IP.

Quand IP rime avec simplicité

Cette section est liée au chapitre 15. 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 l'en-tête en 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.

Prenons l'exemple du paquet IP reproduit dans la figure A2–1. Pour information, il s'agit d'une réponse écho ICMP d'un serveur de Zeste de Savoir.

Figure A2–1
Paquet IP dont on va calculer la checksum.

Offsets	Octet	0								1								2								3							
Octet	Bit	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_{10} ($8bda_{16}$)																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																															

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 l'instant, on considère qu'elle vaut zéro.

Maintenant, retranscrivons ces valeurs en binaire et faisons abstraction des champs. Cela nous donne la figure A2–2.

Figure A2–2
Le même paquet, représenté en binaire.

Offsets	Octet	0								1								2								3							
Octet	Bit	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	0	0	1	1	1	1	0	0
4	32	1	0	0	0	1	0	1	1	1	1	0	1	1	0	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	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	1	0	0
16	128	1	1	0	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	0

On découpe cela par blocs de 16 bits, qu'on additionne :

- 0100 0101 0000 0000
- 0000 0000 0011 1100
- 1000 1011 1101 1010
- 0000 0000 0000 0000

- `0011 0100 0000 0001`
- `0000 0000 0000 0000`
- `0101 1100 1111 0011`
- `0000 0111 0010 1100`
- `1100 0000 1010 1000`
- `0000 0001 0000 1010`

Vous pouvez réaliser 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` et les `0` deviennent des `1`. Cette opération s'appelle **complément à 1**. Nous obtenons ainsi la valeur suivante :

```
1101 0101 0001 0101
```

On peut aussi l'écrire en hexadécimal : `D51516`. Et voilà notre somme de contrôle de l'en-tête ! La figure A2–3 en donne la preuve avec la capture du paquet d'origine, récupérée avec le logiciel Wireshark :

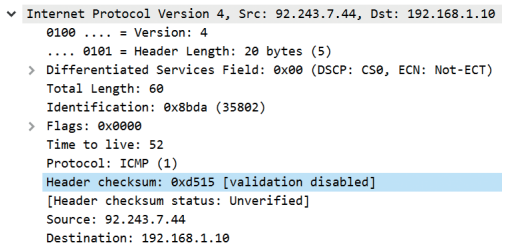


Figure A2–3
La somme de contrôle relevée par analyse de paquet.

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
```

On obtient `1111 1111 1111 1111`. On termine en calculant le 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 !

Nous avons fait le tour de quelques méthodes de calcul 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.

Annexe 3

Analyse fine des communications réseau

Tout au long de ce livre, vous avez pu rencontrer des analyses de trames, qui montrent concrètement les données qui transitent sur le réseau. Elles ont été réalisées au moyen du logiciel Wireshark. Il est utile que vous sachiez vous aussi utiliser un tel analyseur pour 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.

Présentation générale

Le principal intérêt de ces logiciels est de visualiser les trames qui transitent par les interfaces réseau d'un appareil. Cela est utile pour comprendre et résoudre un dysfonctionnement, mais c'est aussi très instructif pour connaître finement un protocole. On peut aussi s'en servir pour 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` (un format de fichier permettant de stocker des trames réseau), 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 la page de téléchargement pour l'installer.

https://www.wireshark.org/#download

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 l'analyse, on lance tout simplement la commande `tcpdump`. Pour l'arrêter, on utilise la combinaison *CTRL+C* sous Windows ou *⌘+C* sous Mac OS. La capture suivante a été prise complètement au hasard.

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens3, link-type EN10MB (Ethernet), capture size 262144 bytes
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
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
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)
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)
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)
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)
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)
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)
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
12:58:39.857156 IP 105.ip-71-57-81.eu > netmon-1-gra.ovh.net: ICMP echo reply,
id 49924, seq 1, length 12
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)
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)
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
```

On peut voir une série de lignes toutes structurées de la même manière :

Heure (au millionième de seconde)	Protocole de niveau 3	Source	Destination	Détails sur le protocole de niveau supérieur
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
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
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)				

Si on veut plus de détails sur ces paquets, on peut ajouter l'option `-vv` à la commande.

```
# tcpdump -vv
tcpdump: listening on ens3, link-type EN10MB (Ethernet), capture size 262144 bytes
13:00:54.067669 IP (tos 0x8, ttl 5, id 1, offset 0, flags [DF], proto ICMP (1),
length 32)
    netmon-1-bhs.ovh.ca > 105.ip-71-57-81.eu: ICMP echo request, id 43960, seq 1,
length 12
13:00:54.067736 IP (tos 0x8, ttl 64, id 10606, offset 0, flags [none], proto ICMP
(1),
length 32)
    105.ip-71-57-81.eu > netmon-1-bhs.ovh.ca: ICMP echo reply, id 43960, seq 1,
length 12
13:00:54.068731 IP (tos 0x0, ttl 64, id 7857, offset 0, flags [DF], proto UDP (17),
length 71)
    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)
13:00:54.070665 IP (tos 0x0, ttl 55, id 52876, offset 0, flags [none], proto UDP
(17),
length 103)
    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)
13:00:54.070967 IP (tos 0x0, ttl 64, id 7858, offset 0, flags [DF], proto UDP (17),
length 71)
    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)
13:00:54.073168 IP (tos 0x0, ttl 55, id 40370, offset 0, flags [none], proto UDP
(17),
```

```
length 104)
    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)
13:00:54.074404 IP (tos 0x0, ttl 64, id 7859, offset 0, flags [DF], proto UDP (17),
length 72)
    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)
13:00:54.076506 IP (tos 0x0, ttl 55, id 40371, offset 0, flags [none], proto UDP
(17),
length 98)
    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)
^C
8 packets captured
8 packets received by filter
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 de UDP.

On peut enregistrer une capture dans un fichier avec l'option `-w` : par exemple, `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 l'article suivant de Lea Linux :

https://lea-linux.org/documentations/Tcpdump

Utilisation de Wireshark

Wireshark est l'outil le plus simple à utiliser pour analyser les trames 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.

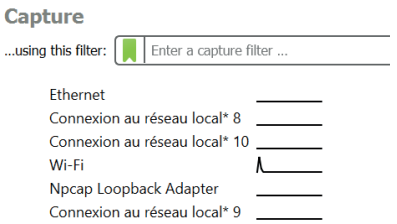


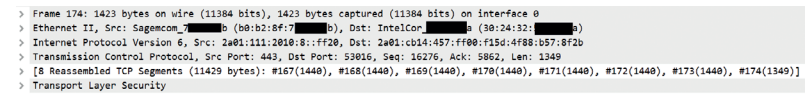
Figure A3–1
Interfaces disponibles dans Wireshark.

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, référez-vous à la documentation officielle. Vous y apprendrez à définir des filtres sur des sous-réseaux (ex. `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 de la figure A3–2 est tirée d'un autre chapitre de ce livre.

Figure A3–2
Informations principales d'une trame, couche par couche.



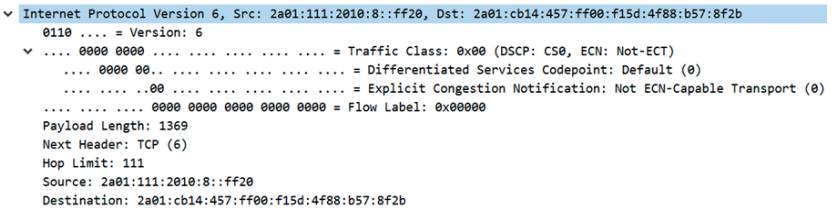
Si vous réalisez une analyse sur une interface sans fil, vous allez vite vous rendre compte qu'il y a anguille sous roche avec la couche 2 : Wireshark montre un autre protocole ! Il s'agit généralement d'Ethernet ou SLL *(Linux cooked)*. Cela est dû au fait que Wireshark n'est pas en mesure de lire réellement 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.

https://wiki.wireshark.org/SLL

Il est techniquement possible de visualiser des trames Wi-Fi avec Wireshark si l'on écoute une interface en mode ***monitor***. C'est trop complexe pour être expliqué ici, mais vous pouvez vous référer à la page suivante (en anglais) : https://wiki.wireshark.org/CaptureSetup/WLAN.

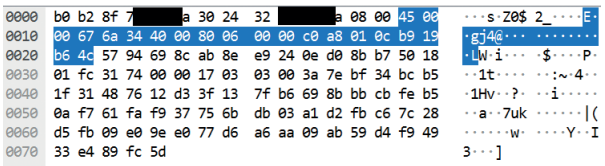
Vous pouvez afficher le détail d'une couche en double-cliquant dessus et faire de même pour chaque section qui débute par un symbole > (figure A3–3).

Figure A3–3
Détail du protocole d'une couche.



Tout en bas, vous trouvez une représentation en hexadécimal de la section sélectionnée. Dans la figure A3–4, 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.

Figure A3–4
Représentation en hexadécimal et en ASCII.



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 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 vous en souvenir.

Annexe 4

Qui gouverne Internet ?

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.

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 *(Internet Corporation for Assigned Names and Numbers)*.

Il s'agit d'une société à but non lucratif basée aux États-Unis, 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. Elle affecte des blocs d'adresses à des **registres Internet régionaux** *(Regional Internet Registry* – RIR*)*. Chacun des 5 RIR est affecté à une région du monde :

- RIPE NCC : Europe, Proche et Moyen-Orient, Russie ;
- APNIC : reste de l'Asie, Océanie ;
- AfriNIC : Afrique ;
- ARIN : Canada, États-Unis et certaines îles proches ;
- LACNIC : reste de l'Amérique.

La figure A4–1, issue de l'Atelier Graphique de Wikipédia et distribuée sous licence CC BY SA 3.0, représente ces différentes plaques géographiques.

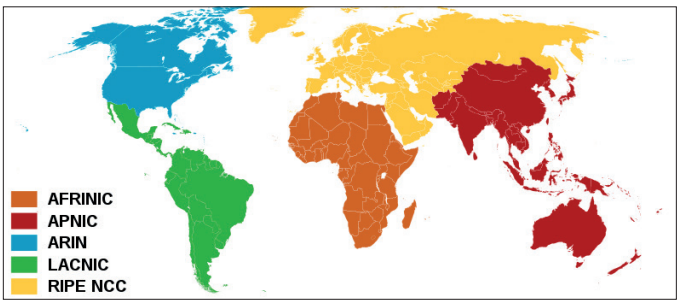


Figure A4–1
Plaques géographiques liées aux RIR.
Source : https://commons.wikimedia.org/wiki/File:Regional_Internet_Registries_world_map.svg?uselang=fr

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, mais délèguent des blocs d'adresses à d'autres entités : les registres Internet locaux *(Local Internet Registry* – LIR*)*. 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** *(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 américaine Verisign a la charge notamment du .com, du .net et du .tv. Les domaines français que sont .fr, .re, .mf, etc., sont gérés par l'AFNIC (Association française pour le nommage Internet en coopération). 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.

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 antarctiques françaises).

Moyennant une redevance, ces différentes entités permettent 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** *(registrars)*. Parmi ces sociétés, on peut citer OVH ou Gandi.

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 *(Internet Engineering Task Force)*. 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* (abrégée ISOC), un organisme à but non lucratif créé par Vint Cerf et Bob Kahn, deux des pères fondateurs d'Internet (inventeurs de TCP/IP). L'ISOC, et l'IETF par là même, fait autorité de facto dans le monde du réseau.

L'IETF édite des RFC *(Requests For Comments)*, 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 obsolète(s). 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 2 119), 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 : https://www.ietf.org/rfc/.

Si on se penche sur l'aspect physique des transmissions réseau, on doit s'intéresser à un autre institut : l'IEEE *(Institute of Electrical and Electronics Engineers)*. 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 1 394 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), `802.1q` (« `dot1q` » quand on parle de VLAN), ou encore `802.3` (Ethernet).

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 !

La figure A4–2 récapitule tout ce que nous avons vu jusqu'à présent.

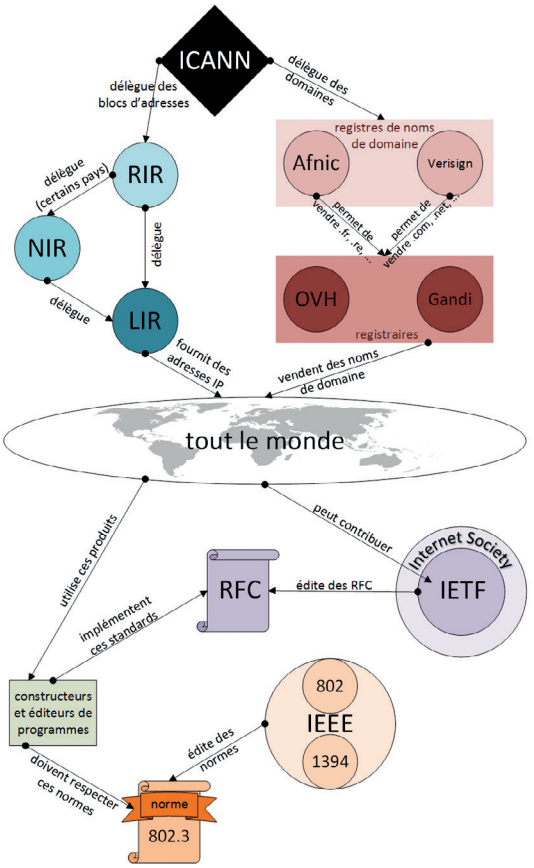


Figure A4–2
Schéma récapitulatif des entités intervenant dans les normes et standards.

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, l'essentiel 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 plate-forme 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.

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 s'appliquer. Nous espérons que vous avez une idée plus claire de qui « gouverne » Internet.

Annexe 5

Packet Tracer, un simulateur de réseau

Le principe d'un réseau, c'est d'avoir plusieurs entités qui communiquent. En informatique, le matériel qui permet tout cela devient vite volumineux et onéreux. Pour pouvoir pratiquer et apprendre à configurer des équipements réseaux sans devoir vendre un rein, il existe des logiciels de simulation. L'un d'entre eux est Packet Tracer, édité par le constructeur de matériel de télécommunications Cisco.

Pourquoi celui-ci et pas un autre, comme GNS3 ? Eh bien, Packet Tracer s'installe sur diverses plates-formes (Windows, Linux, Mac OS), est gratuit, dispose de fonctionnalités avancées, est relativement léger et n'est pas très difficile à prendre en main. Bon, il faut lui reconnaître un point négatif : comme il est édité par l'entreprise Cisco, il ne simule que des équipements Cisco. D'autres compagnies fabriquent du matériel réseau, comme HP ou Alcatel. Le fonctionnement est globalement similaire, mais la prise en main ou la syntaxe des commandes peut être différente.

Bien que nous ayons fait le choix de recourir à un logiciel Cisco, nous n'en faisons pas l'éloge. Nous n'avons aucune forme de partenariat avec cette entreprise et nous ne pouvons que vous encourager à tester d'autres marques, d'autres solutions pour vous faire votre idée. Cela étant dit, il est temps de télécharger ce logiciel.

Téléchargement et installation

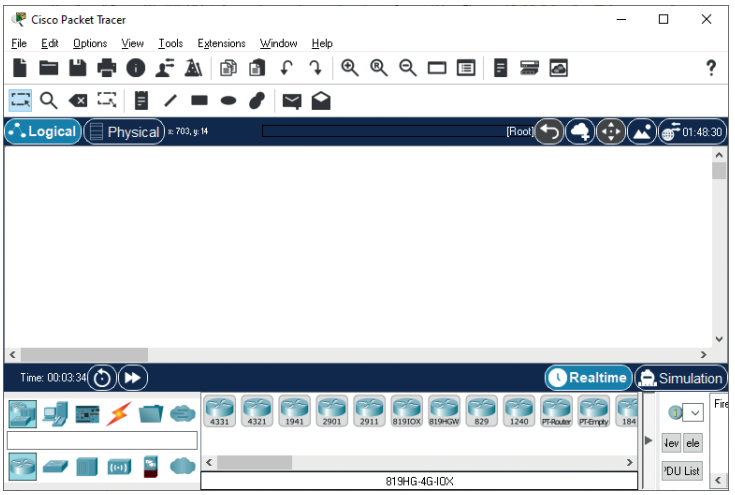
Pour commencer, rendez-vous sur la page de téléchargement de Cisco Packet Tracer (voir encadré ci-après). Vous serez invité à vous connecter ou à créer un compte *NetAcad*. Ce compte sera aussi utilisé pour le premier lancement du programme, alors assurez-vous de retenir vos identifiants.

https://www.netacad.com/portal/resources/packet-tracer

L'installation se fait de la manière habituelle selon votre système d'exploitation. Au premier lancement du programme, lorsqu'il vous est demandé de vous connecter, saisissez les mêmes identifiants que pour le téléchargement. Normalement, c'est la dernière fois que vous en aurez besoin.

Une fois que tout cela est fait, vous devriez arriver sur une fenêtre semblable à la figure A5–1. C'est votre nouveau terrain de jeu, les choses sérieuses peuvent commencer !

Figure A5–1
Interface de Packet Tracer



Interface

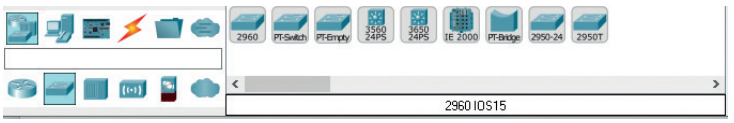
Le moins qu'on puisse dire, c'est que ce logiciel est doté de très nombreuses fonctionnalités. Dans cette annexe, nous allons juste voir comment manier l'interface de manière à pouvoir réaliser les exercices proposés dans le livre.

La grande zone blanche, c'est l'espace de simulation. On peut y placer tous les équipements qu'on veut : routeurs, switchs, serveurs, PC, câbles de différentes sortes…

En dessous, là où se trouvent divers symboles, c'est la zone des équipements. À gauche, vous voyez deux lignes : celle du haut correspond aux catégories générales – équipements réseau, câbles, équipements de terminaison… – tandis que celle du bas désigne des sous-catégories plus spécifiques – routeurs, switchs, hubs… En passant le curseur dessus, vous verrez leur nom apparaître entre ces deux lignes (voir ci-dessous).

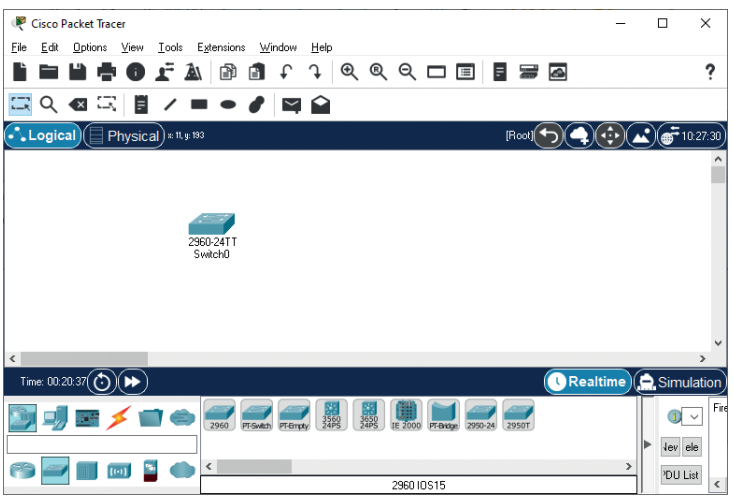
La zone en bas à gauche correspond aux catégories et sous-catégories d'équipements. Lorsque vous choisissez une sous-catégorie, les équipements qui la composent s'affichent dans l'emplacement à droite de cet espace. Sur la figure A5–2, nous avons sélectionné la sous-catégorie *Switches*, qui dévoile une liste des switchs disponibles : *2960*, *PT-Switch*…

Figure A5–2
La sous-catégorie « switches » dévoile une liste de switchs disponibles : 2960, PT-Switch…



Puisque nous sommes dans les switchs, plaçons le modèle *2960* dans l'espace de simulation. Pour cela, on clique simplement une fois sur le modèle concerné, puis une fois à l'emplacement où on souhaite l'intégrer dans l'espace de simulation. On obtient alors un résultat comparable à la figure A5–3.

Figure A5–3
Interface de Packet Tracer avec un équipement placé dans l'environnement de simulation.



Par défaut, l'appareil porte un nom composé du type d'équipement et d'un numéro, en commençant par `0`. Ainsi, le prochain switch à intégrer la simulation s'appellera `Switch1`. Le premier routeur s'appellera `Router0` et ainsi de suite. Il est tout à fait possible de les renommer en cliquant sur leur nom (pas sur l'icône).

N'hésitez pas à placer de nombreux éléments en vrac pour vous faire la main. Pour en supprimer un, cliquez sur le bouton au-dessus de la zone de simulation, puis sur l'équipement ou le câble à retirer. Conservez quand même le switch, nous allons vous montrer comment le configurer.

Configuration des équipements

Cliquez sur l'icône du pauvre switch qui a survécu à vos hasardeuses manipulations sur l'interface de simulation. Une nouvelle fenêtre s'ouvre (figure A5–4).

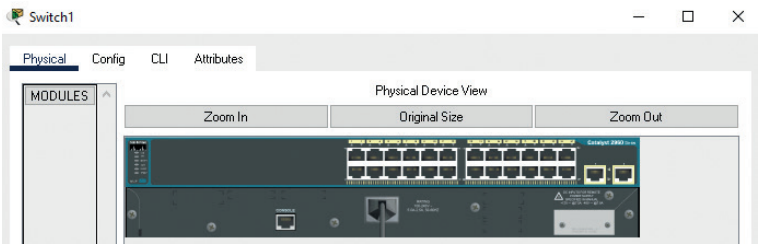


Figure A5–4
Représentation matérielle d'un équipement.

On peut voir quatre onglets : *Physical*, *Config*, *CLI* et *Attributes*. Le seul que nous utiliserons dans les exercices est *CLI (Command Line Interface)*. Oui, les switchs, comme les routeurs, se configurent en ligne de commande.

Il existe un onglet *Config* qui permet de réaliser certaines opérations de configuration en interface graphique. Nous vous interdisons d'y avoir recours tant que vous n'êtes pas à l'aise avec la ligne de commande.

Haut les cœurs, rendez-vous dans l'onglet *CLI* et découvrez un univers austère où tout se fait en tapant des commandes au clavier. Comme vous y invite le système, appuyez sur la touche *Entrée* pour commencer (voir ci-dessous).

```
Press RETURN to get started!


Switch>
```

Ce n'est pas très accueillant tout ça… Néanmoins, vous allez vite réaliser que tout est fait pour que la tâche soit facilitée. Déjà, à chaque fois que vous tapez un point d'interrogation, le système vous dit ce que vous pouvez ou devez saisir, à n'importe quel moment de la commande. Un exemple ? La figure A5–5 montre une liste de commandes que vous pouvez saisir.

Figure A5–5
Taper un « ? » montre les commandes possibles.

```
Switch>?
Exec commands:
  connect     Open a terminal connection
  disable     Turn off privileged commands
  disconnect  Disconnect an existing network connection
  enable      Turn on privileged commands
  exit        Exit from the EXEC
  logout      Exit from the EXEC
  ping        Send echo messages
  resume      Resume an active network connection
  show        Show running system information
  ssh         Open a secure shell client connection
  telnet      Open a telnet connection
  terminal    Set terminal line parameters
  traceroute  Trace route to destination
Switch>
```

On peut lire que `show` donne des informations sur le système. D'accord, mais quelles informations peut-on obtenir ? Tapons donc `show ?` (figure A5–6). On obtient donc la liste des mots-clés pouvant être utilisés après `show` pour afficher des informations. Vous commencez à comprendre le système ? Pour faire défiler les pages et les lignes lorsque `--More--` apparaît, utilisez la barre d'espace.

Figure A5–6
Détail des paramètres possibles de la commande show avec l'aide d'un « ? ».

```
Switch>show ?
  arp                 Arp table
  cdp                 CDP information
  clock               Display the system clock
  crypto              Encryption module
  dtp                 DTP information
  etherchannel        EtherChannel information
  flash:              display information about flash: file system
  history             Display the session command history
  interfaces          Interface status and configuration
  ip                  IP information
  ipv6                Show IPv6 information
  lldp                LLDP information
  mac                 MAC configuration
  mac-address-table   MAC forwarding table
  mls                 Show MultiLayer Switching information
  privilege           Show current privilege level
  sessions            Information about Telnet connections
  ssh                 Status of SSH server connections
  tcp                 Status of TCP connections
  terminal            Display terminal configuration parameters
  users               Display information about terminal lines
  version             System hardware and software status
 --More--
```

Encore plus fort, vous n'avez pas besoin de taper les mots des commandes en entier. Vous pouvez vous contenter de taper les premières lettres puis d'appuyer sur la touche *Tab* de votre clavier pour compléter le mot, ou bien, s'il n'y a aucune autre possibilité de mot qui commence par les mêmes lettres, il est possible de ne saisir que les premières lettres. Ainsi, avec l'habitude, la commande `show history` peut être abrégée en `sh hist`, voire `sh h`. Ces abréviations sont extrêmement courantes, notamment avec les commandes `enable` et `configure terminal` que nous allons voir maintenant.

Privilège et configuration

Le nombre des commandes visibles semble assez limité. En fait, lorsqu'on démarre un switch ou un routeur Cisco, on est en mode `exec`, qui permet uniquement de visualiser certaines informations ou d'effectuer des actions très basiques. Ce mode se traduit visuellement par le symbole > juste après le nom de l'équipement ; dans notre cas, cela donne `Switch>` (figure A5–7). Pour tout le reste, il faut passer en mode privilégié. Pour y accéder, on utilise la commande `enable`, ou `en`. Le chevron devient alors un # (figure A5–7).

Figure A5–7
Le # représente le mode privilégié.

```
Switch>en
Switch#
```

Pour vous rendre compte des nouvelles possibilités qui s'offrent à vous, tapez un point d'interrogation et vous constaterez que d'autres commandes font leur apparition. Parmi elles, `write` permet de sauvegarder en mémoire la configuration de l'équipement. Pensez-y avant de quitter votre travail : même si vous sauvegardez dans l'interface du logiciel au moyen du bouton `Save` , la configuration non enregistrée dans l'équipement est effacée quand il est éteint, c'est-à-dire ici quand vous fermez le programme.

La configuration, justement, est assez limitée en mode privilégié. Pour débloquer toutes les possibilités, il faut passer en mode de configuration avec la commande `configure terminal`, ou `conf t`.

```
Switch#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#
```

C'est de là que vous pouvez paramétrer interfaces, `access-list`, VLAN et autres joyeusetés que vous découvrirez dans ce livre. Pour revenir au mode précédent, par exemple du mode configuration au mode privilégié, tapez la commande `exit`.

Le mode privilégié ne donne pas accès aux mêmes commandes que le mode de configuration ; l'inverse est vrai aussi. Toutefois, pour utiliser une commande telle que `write` alors que vous êtes en pleine configuration d'interfaces, par exemple, vous pouvez la faire précéder du mot `do`. Ainsi, en mode de configuration, vous pouvez sauvegarder vos paramétrages avec la commande `do write`.

Nous avons fait un rapide tour d'horizon de Cisco Packet Tracer. Prenez le temps de vous familiariser avec ce logiciel. N'hésitez pas à faire n'importe quoi avec pour vous faire la main : ce n'est que de la simulation, tout disparaît quand vous fermez le programme. Le matériel physique que l'on trouve dans Packet Tracer fonctionne à l'identique de la simulation. Ainsi, si vous touchez à ces équipements dans le cadre d'une formation, d'études ou de votre travail, vous saurez déjà vous en servir.

Annexe 6

Liste des sigles

ABR	*Area Border Router*
ADSL	*Asymmetric Digital Subscriber Line*
AFNIC	Association française pour le nommage Internet en coopération
API	*Application Programming Interface*
APIPA	*Automatic Private IP Addressing*
ARP	*Address Resolution Protocol*
AS	*Autonomous System*
ASBR	*Autonomous System Boundary Router*
ASCII	*American Standard Code for Information Interchange*
ATM	*Asynchronous Transfer Mode*
BGP	*Border Gateway Protocol*
bit	*binary digit*
CHAP	*Challenge-Handshake Authentication Protocol*
CIDR	*Classless Inter Domain Routing*
CSMA/CA	*Carrier Sense Multiple Access/Collision Avoidance*
CSMA/CD	*Carrier Sense Multiple Access/Collision Detection*
CTS	*Clear To Send*
cwnd	*congestion window*
CWR	*Congestion Window Reduced*

DHCP	*Dynamic Host Configuration Protocol*
DMZ	*DeMilitarized Zone*
DNS	*Domain Name System*
DNSSEC	*Domain Name System Security Extensions*
DOH	*DNS Over HTTPS*
DOT	*DNS Over TLS*
DSCP	*Differenciated Services Code Point*
DV	*Distance vector*
EBCDIC	*Extended Binary Coded Decimal Interchange Code*
EBGP	*External Border Gateway Protocol*
ECE	*ECN Echo*
ECN	*Explicit Congestion Notification*
EGP	*Exterior Gateway Protocol*
EIGRP	*Enhanced Interior Gateway Routing Protocol*
FAI	Fournisseur d'accès à Internet
FQDN	*Fully Qualified Domain Name*
FTP	*File Transfer Protocol*
HDLC	*High-level Data Link Protocol*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IANA	*Internet Assigned Numbers Authority*
IBGP	*Internal Border Gateway Protocol*
ICANN	*Internet Corporation for Assigned Names and Numbers*
ICMP	*Internet Control Message Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IGP	*Interior Gateway Protocol*
IGRP	*Interior Gateway Routing Protocol*
IHL	*Internet Header Length*
IMAP	*Internet Message Access Protocol*
IP	*Internet Protocol*
IPAM	*IP Address Management*
IRC	*Internet Relay Chat*

IS-IS	*Intermediate System to Intermediate System*
ISN	*Initial Sequence Number*
ISOC	*Internet SOCiety*
L2TP	*Layer 2 Tunneling Protocol*
LAN	*Local Area Network*
LIR	*Local Internet Registry*
LSA	*Link-State Advertisement*
LSU	*Link-State Update*
MDA	*Mail Delivery Agent*
MITM	*Man In The Middle Attack*
MLT	*Multi Level Transmit*
MSA	*Mail Submission Agent*
MSL	*Maximum Segment Lifetime*
MSS	*Maximum Segment Size*
MTA	*Mail Transfer Agent*
MUA	*Mail User Agent*
MX	*Mail Exchange*
NAT	*Network Address Translation*
NDP	*Neighbor Discovery Protocol*
NIR	*National Internet Registry*
NRZ	*Non Return to Zero*
NS	*Name Server*
NTLM	*NT LAN Manager*
OSI	*Open Systems Interconnection*
OSPF	*Open Shortest Path First*
P2P	*Peer-to-Peer*
PAP	*Password Authentication Protocol*
PAT	*Port Address Translation*
PC	*Personal Computer*
PDU	*Protocol Data Unit*
PMTUD	*Path Maximum Transmission Unit Discovery*
POP	*Post Office Protocol*
PPP	*Point to Point Protocol*

PPPoA	*PPP over ATM*
PPPoE	*PPP over Ethernet*
PPPoEoA	*PPP over Ethernet over ATM*
QoS	*Quality of Service*
RFC	*Request For Comments*
RIP	*Routing Information Protocol*
RIR	*Regional Internet Registry*
RTO	*Retransmission Time Out*
RTS	*Ready To Send*
RTT	*Round Time Trip*
SDU	*Service Data Unit*
SIP	*Session Initiation Protocol*
SMTP	*Simple Mail Transfer Protocol*
SOA	*Start of Authority*
SOCKS	*Secured Over Credential-based Kerberos*
SPOF	*Single Point of Failure*
SSH	*Secured SHell*
SSL	*Secure Socket Layer*
ssthresh	*Slow start threshold*
TCP	*Transmission Control Protocol*
TD	Travail dirigé
TFTP	*Trivial FTP*
TLD	*Top Level Domain*
TLS	*Transport Layer Security*
TTL	*Time To Live*
UDP	*User Datagram Protocol*
VD	Vecteur de distance
VLAN	*Virtual Local Area Network*
VLSM	*Variable Length Subnet Mask*
VoIP	*Voice over IP*
VPN	*Virtual Private Network*
VRRP	*Virtual Router Redundancy Protocol*
WAN	*Wide Area Network*