Synthex
Informatique

Synthèse de cours et exercices corrigés

Architecture des réseaux

2e édition

Corrigés des exercices

Danièle Dromard

Dominique Seret

PEARSON
Education

Table des matières

Auteurs

Danièle DROMARD, anciennement maître de conférences à l'université Pierre et Marie Curie (Paris 6), est actuellement vacataire chargée de cours en écoles d'ingénieurs. Son domaine d'enseignement et de recherche concerne les architectures informatiques et les réseaux. Elle a publié plusieurs ouvrages sur les réseaux informatiques, dont *Réseaux et télématique, Réseaux informatiques, cours et exercices et L'Architecture SNA*.

Dominique SERET, professeur à l'université Paris Descartes, a dirigé l'UFR (Unité de Formation et de Recherche) en mathématiques et informatique. Elle est responsable du master professionnel MIAGE (Méthodes Informatiques Appliquées à la Gestion des Entreprises). Elle enseigne la logique, l'algorithmique et l'introduction aux réseaux en licence d'informatique ainsi que la sécurité des réseaux en master MIAGE ou en master de recherche en informatique. Passionnée par la pédagogie, elle a participé à plusieurs expériences d'enseignement à distance. Son domaine de recherche concerne plus particulièrement les réseaux et l'évaluation de leurs performances. De nombreuses thèses ont été soutenues sous sa direction. Elle a publié plusieurs ouvrages sur les réseaux, dont Réseaux et télématique, *Réseaux informatiques, cours et exercices, et Introduction aux réseaux*.

Chapitre 1

Les transmissions et les supports

Un réseau suppose plusieurs équipements informatiques (ordinateurs fixes ou portables, divers équipements électroniques, téléphones, assistants numériques personnels…) situés à distance les uns des autres. La première chose à mettre en œuvre pour constituer le réseau est la transmission des informations d'un équipement à l'autre : on utilise des supports de transmission dont nous présentons les caractéristiques dans les deux premières sections. À chaque nature de support correspond une forme particulière du signal qui s'y propage. Il faut fabriquer les signaux, grâce à l'équipement appelé modem. Les techniques de transmission et l'interface entre ordinateur et modem sont normalisées pour assurer l'interopérabilité des équipements. Enfin, nous décrivons brièvement le raccordement ADSL.

Problèmes et exercices

Exercice 1 : notion de décibel

Solution

1. La bande de motards produit huit fois plus de puissance sonore qu'une seule moto. On a : $10 \times \log_{10}(8S) = 10 \times \log_{10}8 + 10 \times \log_{10}S$, ce qui revient à ajouter 10 fois le logarithme décimal de 8 au bruit d'une moto pour obtenir le nombre de décibels produit par les huit motos.

   Puisque : $10 \times \log_{10}8 = 10 \times \log_{10}2^3 = 3 \times 10 \times \log_{10}2 = 9$ dB, la puissance des huit motos vaut : $S = 87 + 9 = 96$ dB.

2. Cela correspond à une puissance sonore de 4×10^9, soit 4 milliards de fois le fond sonore de référence !

Remarque

Pendant que la valeur en décibels du bruit a augmenté d'environ 10 %, la puissance sonore réellement émise a été multipliée par 8.

Exercice 2 : évaluation d'un rapport signal/bruit (S/B)

Solution

1. Un rapport *S/B* de 400 correspond à $10 \times \log_{10}400$: $10 \times (\log_{10}4 + \log_{10}100)$ et $20 \times (\log_{10}2 + \log_{10}100) = 26$ dB.
2. Le rapport *S/B* est 100 fois plus élevé que le précédent, c'est-à-dire qu'il vaut : $26 + 20 = 46$ dB.
3. On peut calculer simplement une bonne valeur approchée du nombre *N* de décibels en remarquant que : $500\,000 = 10^6/2$. On aura donc : $N = 10 \times (\log_{10}10^6 - \log_{10}2) = 10 \times [6 \times \log_{10}10 - \log_{10}2] = 60 - 3 = 57$ dB.

Exercice 3 : débit binaire et rapidité de modulation

Solution

1. D'après la formule $D = R \log_2 V$, on trouve : $D/R = \log_2 V$ soit : $V = 2^{D/R}$; la valence vaut 16.
2. En appliquant la même formule, on trouve : $D = 2\,400 \times 4 = 9\,600$ bit/s.

Exercice 4 : signaux transmis en bande de base et par modulation

Solution

1. Les figures 1.1 et 1.2 représentent les données codées en NRZ et Manchester.

Figure 1.1
Codage NRZ.

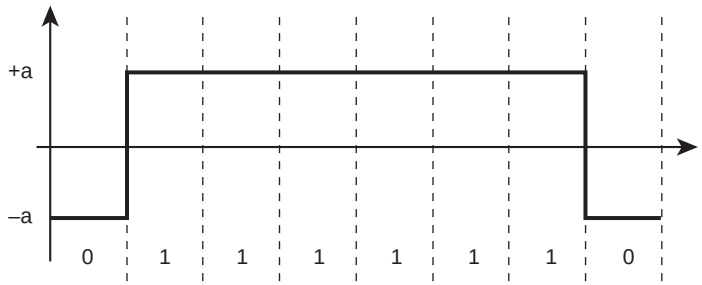
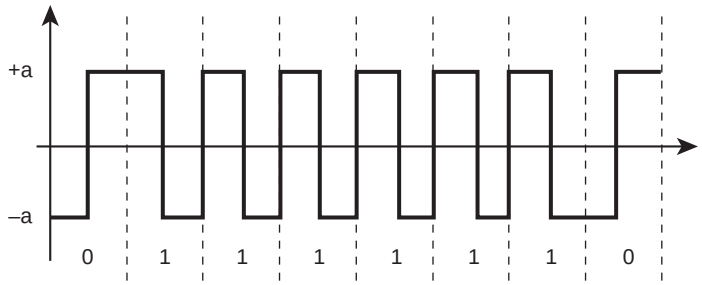


Figure 1.2
Codage biphase ou Manchester.



2. Les modulations d'amplitude et de fréquences sont représentées à la figure 1.3.

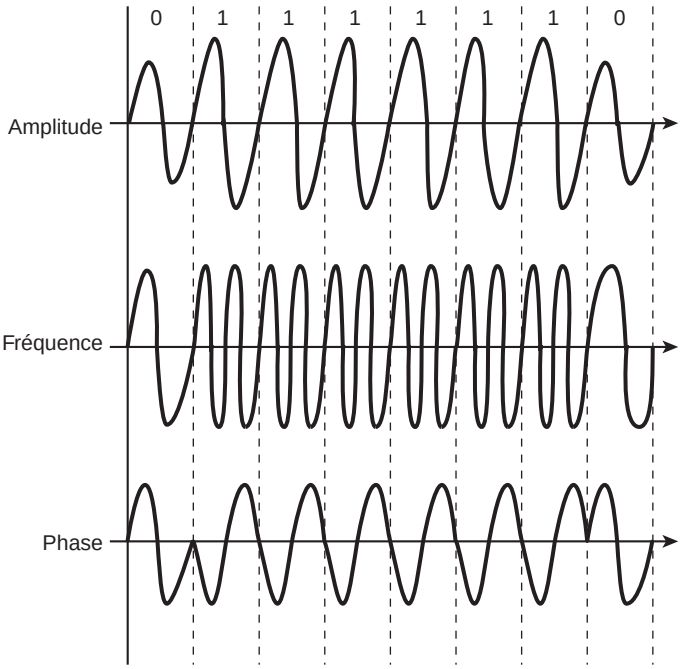


Figure 1.3
Représentation des différentes modulations.

3. Si D est connu et que la valence des signaux soit égale à 2, alors $R = D$ bauds.

Exercice 5 : code Manchester et autres codes

Solution

1. La figure 1.4 représente les données avec le code Manchester.

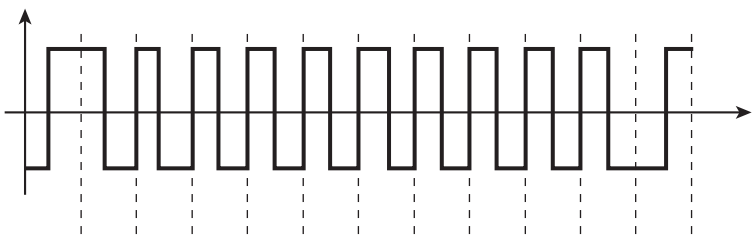
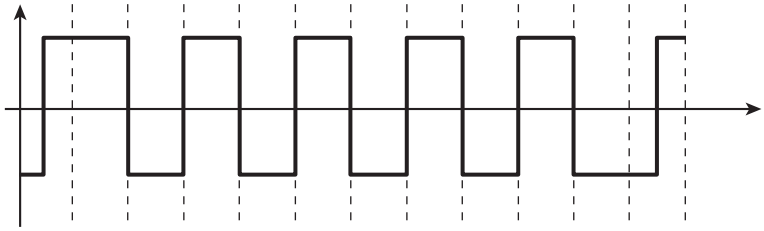


Figure 1.4
Données en codage Manchester.

2. La figure 1.5 représente les données avec le code de Miller.

Figure 1.5
Données en codage de Miller.



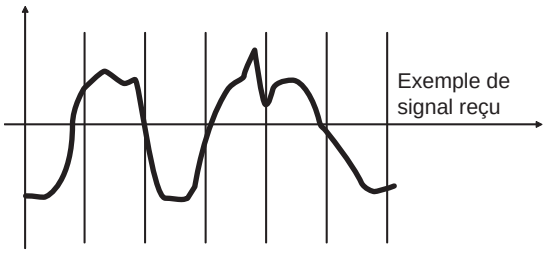
Le décodage du code de Miller est très simple : une transition au milieu de l'intervalle représente un 1, une absence de transition dans l'intervalle représente un 0. Il n'existe aucune ambiguïté de décodage.

Exercice 6 : influence de la phase sur la réception

Solution

1. La figure 1.6 représente les données émises et reçues.

Figure 1.6
Données émises et reçues.



2. On constate que le déphasage a provoqué un mauvais décodage de la suite, puisque la comparaison à la valeur seuil ne s'effectue pas au bon moment.

Remarque

Le choix d'un « bon » code est difficile ! Il faut trouver un compromis entre le nombre de transitions indispensable à la synchronisation du codec récepteur et une solution transparente aux données transmises. Bien évidemment, un tel déphasage du codec récepteur est improbable. Le décalage de phase est particulièrement gênant dans la transmission des données et doit être contrôlé. L'oreille humaine y est très peu sensible.

Exercice 7 : formule de Shannon

Solution

1. On utilise la formule $D = R \times \log_2 V$.

   On obtient : $64 \times 10^3 = R \times \log_2 32$, ce qui donne $D = 5R$, d'où : $R = 12\,800$ bauds. La bande passante est donc égale à 6 400 Hz.

2. En utilisant la formule de Shannon $D = W \times \log_2(1 + S/B)$, on trouve : $64 \times 10^3 = 6\,400 \times \log_2(1 + S/B)$, d'où : $\log_2(1 + S/B) = 10$, c'est-à-dire que $S/B = 2^{10} - 1$, soit 1 023 (on pourra négliger le 1 devant le rapport S/B), ce qui correspond à 30 dB environ.

Exercice 8 : caractéristiques de ligne et téléchargement

Solution

1. Le débit binaire de la ligne vaut 49 600 bit/s. D'après le théorème de Shannon, on obtient : $49\,600 = 3\,100 \times \log_2(1 + S/B)$, soit : $\log_2(1 + S/B) = 16$, d'où : $S/B = 216 - 1$. En négligeant le 1, on trouve un rapport $S/B = 65\,536$, soit environ 48 dB.
2. Toujours en utilisant le théorème de Shannon, on trouve : $24\,800 = 3\,100 \times \log_2(1 + S/B)$, soit : $S/B = 28 - 1 = 255$. Le rapport S/B vaut environ 24 dB.
3. Selon le critère de Nyquist, la rapidité de modulation maximale est égale à deux fois la bande passante de la ligne. Cette dernière vaut donc 2 400 Hz.
4. Le temps t nécessaire pour transférer 2×10^6 octets est égal à : $t = 2 \times 8 \times 10^6/49\,600 = 322{,}58$ s, soit environ 5 minutes et 22 secondes.
5. Le temps t nécessaire n'est plus que de 1,6 s…

Exercice 9 : système de radiomessagerie

Solution

1. Le débit binaire réellement utilisé est : $D = 3\,125 \times 2 = 6\,250$ bit/s.
2. Il faut : $8 \times 200/6\,250 = 0{,}256$ s pour transférer le message sur le récepteur.
3. La bande passante du support vaut : $(169{,}8 - 169{,}425) \times 10^6 = 375$ kHz. D'après le théorème de Shannon, on pourrait transmettre au maximum : $D = 375 \times 10^3 \times \log_2(1 + S/B)$ soit environ : 9 467 495 bit/s.
4. Parce que la vitesse d'affichage utilisée est bien suffisante pour un lecteur humain, puisqu'un écran entier s'affiche en un quart de seconde. On peut ainsi se contenter d'employer des composants bon marché pour la fabrication des récepteurs.

Exercice 10 : principes de fonctionnement de l'ADSL

Solution

1. Il reste 248 canaux pour les flux de données montant et descendant.
2. Le nombre de canaux affectés à chaque sens dépend du débit binaire que l'on veut offrir aux abonnés : plus ce nombre est grand et plus le débit binaire sera important pour le flux considéré. C'est bien évidemment le fournisseur d'accès qui répartit les canaux, en allouant généralement 90 % des canaux au flux descendant et les 10 % restants au flux montant.
3. Il faut simplement allouer autant de canaux pour le flux montant que pour le flux descendant. On obtient ainsi une technologie DSL symétrique (SDSL).
4. On peut obtenir : $4\,312{,}5 \times 32 = 138$ kbit/s pour le flux montant.
5. Il reste pour le flux descendant : $248 - 32 = 216$ canaux, soit un débit binaire de 931,5 kbit/s.
6. On peut obtenir : $15 \times 4\,000 \times 224 = 13{,}44$ Mbit/s.

Remarque

Les technologies symétriques sont réservées aux opérateurs et aux fournisseurs d'accès. Elles ne sont pas disponibles pour les abonnés. On n'atteint pas dans la pratique le débit obtenu à la question 6, car le rapport *S/B* des boucles locales est le plus souvent insuffisant. On obtient couramment 8 Mbit/s sur de courtes distances, avec une boucle locale de bonne qualité.

Chapitre 2

Protocole de communication et contexte de connexion

Dans un environnement où les informations peuvent être altérées, un *protocole de communication* gère les échanges. Celui-ci définit un ensemble de règles, spécifie le format des données et leur délimitation, les moyens de contrôler leur validité, ainsi que le mode de correction des erreurs détectées. Il fixe les modalités du dialogue et fournit en option deux fonctions importantes : le contrôle de flux (contrôle du rythme d'envoi) et la gestion des acquittements (contrôle de la réception des données). Les informations nécessaires aux options sont gérées et stockées dans un *contexte de connexion,* négocié avant le transfert des données. Un protocole sans contexte de connexion assure un service minimal.

Deux équipements directement reliés exploitent un *protocole de liaison*. S'ils sont reliés à travers plusieurs réseaux, le protocole est un *protocole de transport,* dont les fonctionnalités sont les mêmes.

Ce chapitre décrit PPP (*Point to Point Protocol*), la version très simplifiée d'HDLC (*High level Data Link Control,* le protocole de liaison normalisé par l'ITU) pour les accès à Internet. Nous étudierons les protocoles de transport au chapitre 8.

Problèmes et exercices

Exercice 1 : détection d'erreur par VRC et LRC

Solution

1. Il faut ajouter, à chaque caractère, le VRC qui lui correspond puis calculer le LRC du bloc de données. Le tableau 2.1 récapitule les résultats.

Tableau 2.1 : VRC et LRC de la question 1

Données	Codage	VRC
2	0 0 1 0	1
B	1 0 1 1	1
E	1 1 1 0	1
3	0 0 1 1	0
LRC	0 1 0 0	1

On envoie : LRC 3 E B 2, soit dans l'ordre d'émission : 01001 00110 11101 10111 00101.

2. Le bit erroné est indiqué en gras au tableau 2.2 : le récepteur vérifie la parité de chaque donnée. Ici, le quatrième bloc n'est pas correct ; le récepteur refait le calcul du LRC (dernière ligne du tableau) en incluant l'ensemble des données reçues, y compris leurs VRC et LRC. Son résultat fait apparaître une donnée dont la parité n'est pas correcte : le message reçu est rejeté.

Tableau 2.2 : Corrigé de la question 2

	Codage	VRC reçu	Parité de la donnée	Données décodées
	0 0 1 0	1	OK	2
	1 0 1 1	1	OK	B
	1 1 1 0	1	OK	E
	0 **1** 1 1	0	Erreur	
	0 1 0 0	1	OK	

Exercice 2 : VRC/LRC et contrôle polynomial

Solution

1. Le calcul du LRC est donné au tableau 2.3.

Tableau 2.3 : LRC de la question 1

Octet 1	00110011
Octet 2	11110011
LRC	11000000

2. La forme polynomiale du LRC est : $LRC(x) = x^7 + x^6$.
3. Le polynôme $M(x)$ du message est égal à : $x^{13} + x^{12} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x + 1$. Il faut diviser le polynôme $P(x) = x^8 \times M(x)$ par $x^8 + 1$, c'est-à-dire :

$$(x^{21} + x^{20} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^9 + x^8)/(x^8 + 1) = x^7 + x^6$$

Les deux méthodes de calcul donnent le même résultat.

Exercice 3 : calcul d'un contrôle polynomial

Solution

1. Le polynôme $M(x)$ correspondant au message est égal à $x^{13} + x^{12} + x^{11} + x^9 + x^4 + x^2 + 1$. Multiplions-le par x^5, ce qui donne :

$$P(x) = x^5 \times M(x) = x^{18} + x^{17} + x^{16} + x^{14} + x^9 + x^7 + x^5$$

Le reste $R(x)$ vaut $x^4 + x^2 + x + 1$. Le mot de code émis est :

$$P(x) = x^{18} + x^{17} + x^{16} + x^{14} + x^9 + x^7 + x^5 + x^4 + x^2 + x + 1$$

2. Le polynôme $M(x)$ correspondant au mot de code reçu vaut : $x^{16} + x^{14} + x^9 + x^7 + x^5 + x + 1$. Il n'est pas identique au mot de code émis de la question 1. Effectivement, la division polynomiale donne un reste non nul, valant :

$$R(x) = x^4 + x^2 + 1$$

Le récepteur ignore donc le bloc de données.

Remarque

Dans ce bloc de données, on constate que plusieurs bits ont été mal transmis ; sinon, le polynôme reçu serait identique à celui trouvé à la question 1. Le récepteur ne connaissant évidemment pas le bloc émis, il prend sa décision sur le bloc reçu. Remarquons que les erreurs résidaient aussi bien dans le corps du message que dans le bloc de contrôle, et le récepteur ne pouvait pas le savoir !

Exercice 4 : contrôle polynomial avec le polynôme V41

Solution

1. $M(x) = x^7 + x^5 + x^3 + x^2 + x$.
2. La division polynomiale à effectuer est : $x^{16} \times M(x)$ à diviser par $G(x)$, soit : $x^{23} + x^{21} + x^{19} + x^{18} + x^{17}$ à diviser par $x^{16} + x^{12} + x^5 + 1$. Le reste est : $R(x) = x^{14} + x^{12} + x^{10} + x^5 + x^2$, soit en binaire un bloc FCS sur 16 bits : `01010100 00100100`.

Exercice 5 : contrôle d'erreur dans TCP

Solution

1. Le résultat de l'addition de blocs de 16 bits modulo 2 tient sur 16 bits.
2. Le message TCP peut être transcrit en binaire et découpé en blocs de 16 bits (sans écrire les deux derniers qui ne contiennent que des bits à 0 et n'interviennent donc pas dans le calcul de l'addition finale) :

   `00 15 = 0000 1000 0001 0101` ;

   `0F 87 = 0000 1111 1000 0111` ;

   `9C CB = 1001 1100 1100 1011` ;

   `7E 01 = 0111 1110 0000 0001` ;

   `27 E3 = 0010 0111 1110 0011` ;

   `EA 01 = 1110 1010 0000 0001` ;

   `50 12 = 0101 0000 0001 0010`.

   L'addition fournit `0111 0000 1011 1000`, soit en hexadécimal `E0 B8`.

Exercice 6 : circuit de calcul du bloc de contrôle d'erreur

Solution

1. Le registre étant initialisé à plein 0, il contient le premier octet lui-même une fois que les huit premiers bits sont entrés. Le premier bit se retrouve dans la case 7, le deuxième dans la case 6, etc.
2. Au neuvième top d'horloge, se présente à l'entrée Data in le premier bit du deuxième octet, alors que sort à Data out le premier bit du premier octet. Ces 2 bits sont additionnés modulo 2 (OU exclusif), et le résultat est rangé dans la case 0, laissée libre par le décalage. Une fois que toutes les données sont entrées, on a donc l'addition bit à bit des 2 octets. On a vu à l'exercice 3 qu'il s'agissait du LRC et du reste de la division du polynôme associé aux données par $x^8 + 1$. Le circuit de la figure 2.9 du livre est donc le circuit de calcul du LRC.

Remarque

Quel que soit le polynôme diviseur, un simple circuit exécute le calcul de la division *à la volée*. Il comprend un registre à décalage, avec autant de cases que le degré du polynôme et autant d'opérateurs OU exclusif qu'il existe de termes non nuls dans ce polynôme. Le registre est initialisé à 0. Dès que toutes les données sont entrées, il contient le reste de la division. L'émetteur n'a plus qu'à faire sortir bit à bit le champ de redondance pour l'insérer à la suite des données. Avec un tel circuit, le calcul s'effectue en série dès l'arrivée du bit (inutile de connaître la totalité des données avant de commencer). Le résultat est instantané, alors que le LRC se calcule en parallèle sur tous les bits du message. Le même circuit sert aussi chez le récepteur puisque celui-ci refait la même division.

Exercice 7 : échange de données par satellite

Solution

1. Soit T le temps de transmission, l la longueur en bits du message, t_p le temps de propagation (temps mis par le signal à la vitesse de la lumière, 300 000 000 m/s) et D le débit binaire en bits par seconde. On a la relation : $T = l/D$.

   A transmet un message à B, qui le reçoit à $T + t_p$ et B envoie immédiatement la réponse. Pour éviter toute perte de temps, il faut que la taille de la fenêtre corresponde à un délai supérieur à T augmenté du temps d'attente de la réponse. Calculons le temps de transmission d'un message et le temps de propagation. Le temps de transmission d'un message à 64 octets de données vaut :

   $$T = (48 + 64 \times 8)/9\,600 = 58{,}33 \text{ ms}$$

   À 200 km d'altitude, le temps de propagation (aller-retour Terre-satellite) vaut :

   $$t_p = 2 \times 200\,000/300\,000\,000 = 1/750 = 0{,}001333333 = 1{,}333 \text{ ms}$$

   Si Δ est le temps d'attente de réception du message de réponse et $t_{réponse}$ son temps de transmission :

   $$\Delta = 2 \times t_p + t_{réponse} = 2\,t_p + (48/9\,600) = 2 \times 1{,}33 + 5 = 7{,}66 \text{ ms}$$

   Pendant qu'on envoie le second message par anticipation, on reçoit la réponse au premier. Il n'y a donc pas de problème de taille de fenêtre.

2. À 36 000 km d'altitude, le temps de propagation devient :
   $t_p = 2 \times 36\,000/300\,000 = 240$ ms. Δ vaut alors :

   $$\Delta = 2 \times t_p + t_{réponse} = 485 \text{ ms}$$

   Soit F le volume de données transmises avant qu'on finisse de recevoir la réponse. On trouve :

   $F = \Delta/D = 80\,000$ bits environ. Il y a un silence chaque fois que la fenêtre est pleine. Si on choisit une taille de fenêtre supérieure à 80 000 bits, on limite les problèmes d'anticipation. Remarquons qu'un tel volume de données tient en 8,3 messages ; donc, si les messages sont numérotés individuellement, il est indispensable que le modulo de la numérotation soit supérieur à 8 ou que les messages envoyés soient plus longs sinon.

Exercice 8 : relation entre fenêtre et modulo de la numérotation

Solution

Soit W la taille de la fenêtre. Si elle est égale à N le modulo de la numérotation, le message de rang k et celui de rang $k + N$ portent le même numéro k, puisque les deux rangs ont le même reste de division par N.

Un cas d'ambiguïté est décrit ci-dessous avec $N = 8$. Les numéros possibles sont : 0, 1, 2… 7 (*Maxseq* = 7). Si $W = 8$, une station qui émet plusieurs messages dont le premier est mal transmis reçoit un acquittement RR0. Si, maintenant, elle envoie huit messages consécutifs avec succès, elle reçoit également RR0 ! La station réceptrice considère que le huitième message est un doublon du message 0 (puisque, pour elle, les deux portent le même numéro). La station réceptrice ignore les huit messages qu'elle a pourtant reçus correctement…

On peut conclure de cet exemple que la taille maximale de la fenêtre doit être au plus égale à *Maxseq*.

Exercice 9 : correction d'erreurs avec numéro de messages

Solution

1. Le protocole décrit est un protocole en mode connecté : il gère un contexte de connexion contenant au moins le numéro du prochain message à émettre (pour le module émission) et celui du prochain message à recevoir (pour le module réception).
2. Les numéros sont dans l'ordre : 0, 1, 2, 3 et 4. La fenêtre doit être supérieure ou égale à 5 pour qu'un émetteur puisse envoyer ces cinq messages, même en l'absence d'accusé de réception du récepteur.
3. Le troisième message (portant le numéro 2) est ignoré par *B*. Le module de réception de *B* voit donc les messages dans l'ordre : 0, 1, 3 et 4. Après avoir reçu le message 1, il s'attend au message 2. *B* constate donc seulement après avoir reçu le message portant le numéro 3 que les numéros ne se suivent pas. Il demande alors à *A* de retransmettre tout depuis le message portant le numéro 2.
4. Si la fenêtre est limitée à deux messages, *A* ne peut envoyer que les messages 0 et 1 et doit s'arrêter pour attendre les accusés de réception. En l'absence d'information sur le délai pour recevoir les accusés de réception, on peut imaginer deux cas : (a) les accusés de réception sont pratiquement instantanés et, même limitée à deux messages, la fenêtre n'est pas bloquante ; (b) l'accusé de réception du premier message n'est pas arrivé avant la fin de l'émission du message portant le numéro 1. *A* est bloqué en attente de cet accusé de réception (voir figure 2.1).

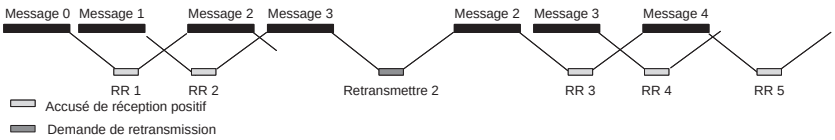


Figure 2.1 Schéma des échanges.

Exercice 10 : correction d'erreurs avec numérotation du flux de données

Solution

1. Le protocole décrit est un protocole en mode connecté : il gère un contexte de connexion qui comprend au moins le numéro du premier octet, le numéro du prochain octet à émettre (pour le module émission) et celui du prochain octet à recevoir (pour le module réception).
2. Les numéros sont dans l'ordre : 45, 345, 645, 945 et 1245, puisqu'il y a 300 octets par message.
3. Le troisième message (portant le numéro 645) est ignoré par *B*. À la réception, *B* voit qu'il a reçu correctement les octets depuis 45 jusqu'à 644 puis un message contenant les octets de 945 à 1244. Quand il examine ce message, il constate l'intervalle manquant et demande à *A* de retransmettre les données à partir de l'octet 645.

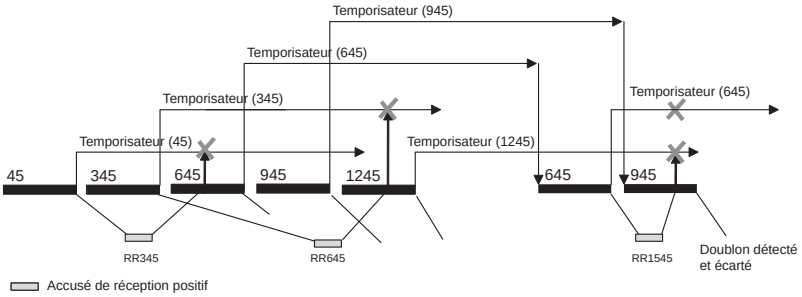
Exercice 11 : stratégie passive du récepteur

Solution

La fenêtre de 2 000 octets n'est pas bloquante puisqu'il n'y a ici que 1 500 octets à transmettre entre *A* et *B*. Avec la stratégie proposée, *A* déclenche un temporisateur à chaque envoi de message et attend l'accusé de réception correspondant. À la fin du schéma de la figure 2.10 du livre, les deux premiers temporisateurs ont été arrêtés ; par contre, les trois suivants courent encore.

La station *B* ne reçoit pas le message 645 qui est perdu dans le réseau. Elle ne fait rien d'autre que conserver en attente les messages 945 et 1245, qui contiennent des données non contiguës aux précédentes. C'est donc *A* qui réagit lorsque ses temporisateurs expirent : celui du message 645 d'abord, qui provoque la retransmission du message 645, puis celui du message 945 qui provoque la retransmission du message 945 (voir figure 2.2).

Figure 2.2
Schéma des échanges.



Pendant ce temps, *B* reçoit le message 645 ; ce dernier vient « boucher le trou » entre les données reçues au début et celles mises en attente. Toutes ces données se suivent ; il peut donc acquitter tout l'ensemble et dire qu'il est prêt à recevoir l'octet 1545. L'accusé de réception provoque l'arrêt, dans la station *A*, de deux temporisateurs : celui du message 1245 et celui du message 645 relancé. Il évite à *A* de relancer le temporisateur du nouveau message 945. Quand *B* le reçoit, il constate que celui-ci contient des données déjà reçues et acquittées ; il s'agit donc d'un doublon : le message est écarté.

Remarque

Cet exercice fait apparaître des délais variables de traversée des réseaux, ce qui est inévitable dans Internet. Les messages pourraient même arriver dans le désordre si les routes empruntées changent.

Chapitre 3

Concepts généraux et modélisation des architectures de réseaux

Si la complexité de l'infrastructure d'un réseau dépend de sa taille, les services et les protocoles mis en œuvre dans un réseau d'opérateur sont actuellement les mêmes que dans le réseau d'une entreprise. Dans les trois premières sections, nous étudions tout d'abord les techniques de commutation et de multiplexage, qui optimisent les coûts de fonctionnement et de maintenance d'un grand réseau. Nous évoquons ensuite les fonctions de contrôle interne utilisées pour gérer au mieux les ressources disponibles et garantir le meilleur usage possible aux utilisateurs d'un réseau.

La quatrième section est consacrée aux trois modèles qui structurent les architectures de communication : le modèle OSI *(Open System Interconnection)* ou modèle de référence, le modèle IEEE *(Institute for Electricity and Electronics Engineers)* défini pour les réseaux locaux et la pile des protocoles TCP/IP *(Transport Control Protocol / Internet Protocol)*. Le premier, plus ancien, a apporté une terminologie et des concepts toujours en usage dans les réseaux ; le dernier est le standard *de facto* sur lequel s'appuient les systèmes de communication actuels.

Enfin, la cinquième section présente les services offerts dans un réseau.

Problèmes et exercices

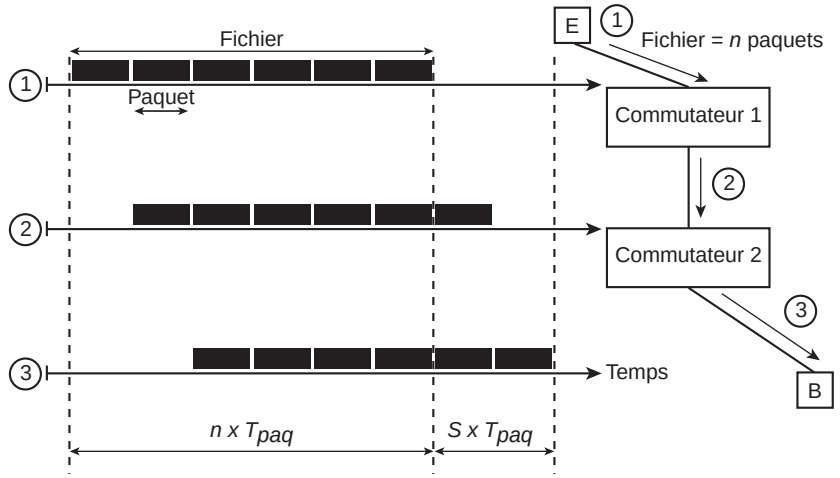
Exercice 1 : temps de transmission dans un réseau à commutation

Solution

1. La durée de transmission du fichier sur une liaison est égale à : $T_{fic} = (L + H)/D$. La durée de transmission du fichier est donc égale au temps de transmission sur toutes les liaisons traversées, c'est-à-dire : $T_{fic1} = T_{fic} \times (S + 1)$.
2. La durée de transmission d'un paquet sur une liaison de données vaut : $T_{paq} = (P + H)/D$. La durée de transmission du fichier est donc égale à la durée de transmission des paquets jusqu'au premier commutateur, plus le délai nécessaire au dernier paquet pour parvenir jusqu'à *B*. Le nombre de paquets nécessaires pour transmettre le fichier vaut $n = L/P$. On en déduit : $T_{fic2} = (S + n) \times T_{paq} = (S + n) \times (P + H)/D$.

   La figure 3.1 montre comment calculer les différents temps de transmission.

Figure 3.1
Calcul des différents temps de transmission.



3. Applications numériques :
   a. Cas de la commutation de messages : $P = L = 640\,008 = 512\,000$ bits ;

   $$T_{fic1} = (2 + 1) \times (64\,000 + 9) \times 8/64\,000 = 24 \text{ s}$$

   b. Cas de la commutation de paquets avec $P = 128$ octets : $P = 128 \times 8 = 1\,024$ bits ; $n = L/P = 500$ paquets ;

   $$T_{fic2} = (2 + 500) \times (128 + 9) \times 8/64\,000 = 8{,}6 \text{ s}$$

   c. Cas de la commutation de paquets avec $P = 16$ octets : $P = 16 \times 8 = 128$ bits ; $n = L/P = 4\,000$ paquets ;

   $$T_{fic2} = (2 + 4\,000) \times (16 + 9) \times 8/64\,000 = 12{,}5 \text{ s}$$

d. Cas de la commutation de cellules ATM avec $P = 48$ octets, $H = 5$ octets : $P = 48 \times 8 = 384$ bits ; $n = L/P = 1\,334$ paquets (par excès) ;

$$T_{fic2} = (2 + 1\,334) \times (48 + 5) \times 8/64\,000 = 8{,}85 \text{ s}$$

Remarque

Avec la commutation de messages, le temps de transmission du fichier ne dépend que du nombre de liaisons traversées. En revanche, avec la commutation de paquets, il faut tenir compte du recouvrement des temps de transmission des différents paquets sur l'ensemble des liaisons : en effet, pendant que *A* transmet son deuxième paquet au premier commutateur, celui-ci envoie le premier paquet au commutateur suivant et ainsi de suite. C'est la raison pour laquelle les performances de la commutation de paquets sont supérieures à celles de la commutation de messages. L'écart des performances sera encore plus grand si certaines liaisons transmettent le message avec des erreurs, comme nous le verrons avec les questions suivantes.

4. Le découpage en paquets réduit les délais d'acheminement à travers le réseau. Cependant, il faut respecter une juste proportion entre la taille de l'en-tête et celle du corps de message : une taille de paquet trop petite provoque un allongement du délai.
5. Pour qu'un message de longueur L soit reçu sans erreur, il faut que tous ses bits soient reçus sans erreur. La probabilité de recevoir 1 bit sans erreur vaut $1 - \tau$. La probabilité de recevoir L bits sans erreur vaut donc : $(1 - \tau)^L$. La probabilité de recevoir un message erroné est donc de $p_t = 1 - (1 - \tau)^L$.

   Puisque la longueur d'une trame vaut : $L = P + H$, le nombre moyen d'émissions est donc :

   $$1 \times (1 - p_t) + 2 \times (1 - p_t)\, p_t + 3 \times (1 - p_t)\, p_t^2 + \ldots = 1/(1 - p_t)$$

   En appliquant la formule précédente et en tenant compte des répétitions, on obtient :

   $$T'_{fic} = T_{fic}/(1 - p_t) = T_{fic}/(1 - \tau)^L$$
6. Les applications numériques donnent :
   a. Cas de la commutation de messages : $P = L = 64\,000 \times 8 = 512\,000$ bits ;

      $T'_{fic} = 16\,848$ s, soit plus de quatre heures !

   b. Cas de la commutation de paquets avec $P = 128$ octets : $P = 128 \times 8 = 1\,024$ bits ;

      $T'_{fic} = 9{,}6$ s, soit une dégradation de 11,6 % par rapport au cas parfait

   c. Cas de la commutation de paquets avec $P = 16$ octets : $P = 16 \times 8 = 128$ bits ; $n = L/P = 4\,000$ paquets ;

      $T'_{fic} = 12{,}75$ s, soit une dégradation de 2 % par rapport au cas parfait

      $T'_{fic} = 9{,}22$ s, soit une dégradation de 4,2 % par rapport au cas parfait
7. La prise en compte du taux d'erreurs dans les liaisons montre tout l'intérêt du découpage des messages en paquets. Il est visiblement hors de question d'utiliser la commutation de messages pour les applications nécessitant de hauts débits, tout particulièrement lorsque les liaisons sont peu fiables. On voit également qu'une taille de paquet trop petite est un choix peu judicieux. Les cellules ATM et les paquets de 128 octets sont donc des compromis intéressants entre les différentes contraintes pour les hauts débits.

Exercice 2 : signalisation dans les multiplexeurs E1

Solution

1. Il faut 15 trames pour transporter la signalisation de toutes les voies *BV*.
2. Il faut attendre 16 trames entre deux signalisations successives de la même voie, soit : $16 \times 125\ \mu s = 2$ ms.

Exercice 3 : structure de trame d'un multiplexeur T1

Solution

1. Longueur totale de la trame *T1* : $24 \times 8 + 1 = 193$ bits.
2. Le temps entre deux trames correspond au temps séparant deux caractères successifs émis sur une voie *BV*. Puisque le débit des voies téléphoniques est 64 kbit/s, il faut 125 µs pour transmettre 1 octet d'une voie *BV*.
3. Il faut transmettre 193 bits en 125 µs, d'où un débit binaire de : $193/125 \times 10^6 = 1{,}544$ Mbit/s. On peut aussi considérer que le multiplexeur émet 8 000 trames par seconde (une trame toutes les 125 µs), ce qui nécessite un débit de : $193 \times 8\ 000 = 1{,}544$ Mbit/s.

Remarque

Le débit de la voie multiplex dépend de plusieurs critères : la manière de transmettre la signalisation et les données, le nombre de lignes à multiplexer et le débit des voies *BV*. Les deux premiers critères conditionnent la structure de la trame multiplex ; le dernier facteur définit le rythme d'occurrence des trames.

Exercice 4 : routage par inondation

Solution

1. La recherche entreprise est la plus rapide possible, car le temps de recherche est proportionnel au nombre de sauts pour atteindre le nœud de rattachement de NT7 et non au nombre total de nœuds du réseau.
2. Une recherche de ce type entraîne une prolifération de messages de requêtes et de réponses dans le réseau et dégrade considérablement ses performances. On ne l'envisage que pour des recherches exceptionnelles, sinon le réseau risque d'être complètement saturé.

3. NR1 sait que la requête est terminée lorsqu'il a reçu la réponse de ses voisins immédiats, ici NR2 et NR3.
4. Pour montrer le travail des différents NR*i*, nous utilisons les conventions suivantes à la figure 3.2 :
   – Une requête émise par un NR*i* vers un autre nœud est représentée par une flèche terminée par un point d'interrogation.
   – La réponse du nœud contacté est représentée par une flèche dont l'extrémité est annotée « non » si l'équipement terminal recherché n'a pas été trouvé dans les nœuds voisins et « oui » si le nœud émetteur de la réponse a trouvé la ressource demandée.

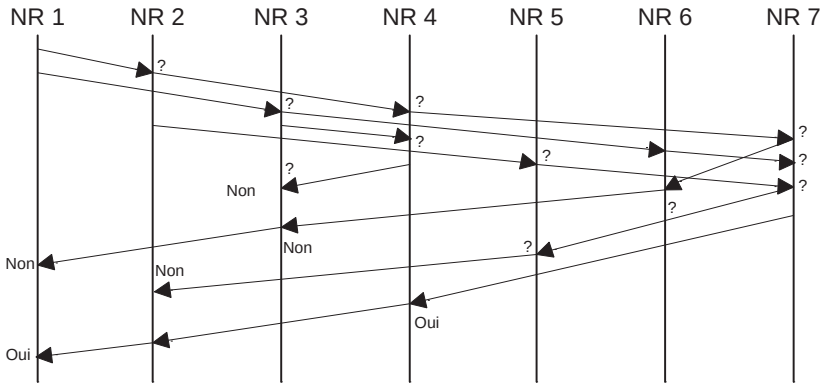


Figure 3.2 Diagramme des requêtes échangées entre les différents nœuds.

Explications et commentaires : dès réception de la requête de NR1, NR2 l'envoie à ses voisins, soit NR4 et NR5. Après avoir reçu une réponse de ses deux voisins, NR2 répond à NR1. NR3 procède de même pour les nœuds NR4 et NR6, et ainsi de suite. Il faut donc trouver un moyen de propager, le plus rapidement et le plus simplement possible, une seule réponse positive vers NR1. Supposons que la requête de NR2 répercutée par NR4 arrive la première à NR7.

Examinons ce qui se passe dans NR4. Celui-ci a répercuté la requête de NR2 vers NR3 et NR7. Entre-temps, NR3 a envoyé la requête vers NR4 et NR6. NR4 peut donc interpréter la requête émanant de NR3 comme une réponse négative à sa propre requête : si NR3 avait trouvé la ressource, il le signalerait directement dans sa réponse à NR1, donc NR4 n'a pas à s'en occuper puisque NR3 sera plus rapide que lui à répondre. De la même manière, quand NR3 reçoit une requête de NR4, il l'interprète comme une réponse négative à sa propre requête : même si NR4 a trouvé la ressource, le fait qu'il envoie la requête à NR3 signifie qu'il a reçu une requête provenant d'un autre chemin et qu'il notifie sa réponse positive sur cet autre chemin plus rapide.

NR7, de son côté, répercute la requête reçue de NR4 vers NR5 et NR6, bien qu'il sache où se trouve la ressource demandée. Là encore, ces deux nœuds interprètent la requête de NR7 comme une réponse négative à leur requête. De cette façon, le chemin le plus rapide pour propager la réponse vers NR1 est privilégié. Ainsi, NR7 envoie « oui » à NR4 et « non » à NR5 et NR6. NR4 répond positivement à NR2. Celui-ci envoie sa réponse à NR1 dès qu'il a reçu la réponse de NR5.

Exercice 5 : choix des primitives d'un niveau donné

Solution

Trois considérations principales doivent guider le concepteur d'un niveau donné :

- définir les services offerts par la couche (i) aux entités ($i + 1$), en tenant compte de la position de ce niveau dans l'architecture de communication ;
- proposer un nombre minimal de primitives différentes, afin de ne pas compliquer l'interface ;
- minimiser le nombre de paramètres à prendre en compte dans chaque primitive définie.

Satisfaire à la première condition nécessite une idée précise du fonctionnement de la couche (i). Par ailleurs, la complexité de l'interface dépend du niveau considéré : une couche de bas niveau offre forcément des services plus limités qu'une couche haute de l'architecture. Il faut donc identifier les services disponibles et décider du nombre d'entités nécessaires à leur gestion, puis déterminer les interactions entre elles, afin de définir la circulation des informations au sein de la couche à spécifier.

La deuxième condition répond à un souci d'efficacité. Si l'interface compte un grand nombre de primitives, l'entité ($i + 1$) risque au mieux de ne pas en utiliser toutes les subtilités. Au pire, elle peut entraîner une baisse des performances, préjudiciable à toute l'architecture. En effet, un nombre élevé de primitives risque fort de mener à des doublons. En outre, les entités ($i + 1$) pourraient ne pas utiliser certaines primitives.

La troisième condition fournit une meilleure lisibilité du service demandé et accélère son traitement. Si une primitive compte 10 paramètres, chacun d'entre eux étant géré par une entité, il faut concevoir 10 entités pour les manipuler, définir leur mode de coopération et prévoir toutes les situations possibles entre chaque paire d'entités concernées par le traitement du service… Cela risque de provoquer des boucles dans le parcours des données entre les entités exécutant le service demandé. Dans tous les cas, le temps de traitement de la primitive s'en trouve notablement allongé. Un compromis est à trouver entre des conditions contradictoires : une primitive simple se traite plus efficacement, mais elle n'exécute qu'un service limité. Augmenter le nombre de primitives est risqué : un excès de primitives conduit forcément à de piètres utilisations…

Exercice 6 : niveaux d'adressage dans un réseau

Solution

1. Puisque la liaison 2 est coupée, les communications sont brutalement interrompues entre M1 et les clients qui utilisent cette liaison. Par contre, rien ne change pour les clients utilisant la liaison 1 : ils continuent d'accéder normalement au serveur M1.

2. L'adresse IP de destination sert à trouver le chemin vers le serveur. Toutefois, la connaissance des adresses IP origine et destination ne peut suffire car un même client peut gérer plusieurs connexions avec le même serveur. Il faut donc connaître les identifiants de connexion utilisés chez le client et chez le serveur. La distinction entre les flots se fera au niveau TCP, grâce à l'identifiant de connexion, unique pour chaque connexion.

   On utilise le concept de *socket* pour identifier localement une connexion[1]. Un socket se compose du doublet : < adresse IP, numéro de port > dans lequel le numéro de port est l'identifiant de l'application (unique dans la machine locale). Le socket local est constitué du doublet < adresse IP locale, numéro de port local >. Le flot de données est identifié par < nom du protocole Transport, socket local, socket distant > (voir chapitre 8).

3. Au niveau IP, M1 ne peut plus envoyer ou recevoir de données. Le protocole IP n'entreprend aucune action de reprise puisque son service est sans garantie. Par contre, le protocole TCP, fonctionnant en mode connecté, détecte une fin anormale de communication. Il réinitialise les connexions brutalement interrompues. Certains éléments de message transmis au moment de la rupture de communication pourront manquer ou être transmis deux fois, mais TCP assure la récupération des éléments manquants et la détection des doublons.

Exercice 7 : procédures en cas de panne du réseau

Solution

1. L'émetteur ignorant la panne survenue dans le réseau, il continue d'envoyer les données vers le destinataire. Il appartient au réseau de communication de détecter la panne et de trouver un chemin alternatif passant par d'autres nœuds pour atteindre le destinataire. Tant que la phase de recherche d'un nouveau chemin n'a pas abouti, les données émises par l'émetteur s'accumulent dans les nœuds situés en amont de la panne. S'ils sont saturés, les données sont irrémédiablement perdues.

Remarque

Cet exemple illustre l'importance de l'efficacité de la fonction de routage dans un réseau de communication offrant un service sans connexion puisque toute panne se traduit par une perte de données.

2. Il convient de distinguer deux cas :
   a. Le service rendu est en mode connecté dans tous les niveaux de l'architecture de communication.
   b. Le service en mode connecté s'appuie sur des couches sous-jacentes fonctionnant en mode sans connexion.

1. Cette notion provient du système UNIX, très largement utilisé dans les universités américaines, à l'origine des protocoles TCP et IP.

Dans le cas a, la panne est détectée par le nœud situé en amont du nœud défaillant. Le nœud qui l'a détectée refuse l'arrivée de nouvelles données et recherche un nouveau chemin vers le destinataire. Dès que ce chemin est trouvé et établi, il accepte de nouveau les données provenant de l'émetteur. Au pire, la perte de données est limitée aux données transmises juste au moment de la panne.

Dans le cas b, puisque le niveau sous-jacent fonctionne en mode non connecté, seul l'équipement terminal destinataire constate qu'il manque tout ou partie des données selon la durée du transfert. Il demande donc à l'équipement terminal émetteur de réémettre les données à partir d'un point de reprise négocié entre les deux extrémités. Cela peut occasionner la duplication de données correctement acheminées au moment de la panne.

Remarque

Le cas a est celui qui entraîne le moins de perte de données, au prix d'une surveillance constante et à tous les niveaux des transferts de données dans le réseau de communication. Cette précaution est superflue et pénalisante pour les performances si le réseau de transport est suffisamment fiable.
Dans le cas b, la gestion correcte du service en mode connecté implique de nombreux contrôles et des procédures de reprise sophistiquées pour assurer un transfert de données satisfaisant entre les deux extrémités.

Exercice 8 : interfaces dans un réseau et qualité de service

Solution

1. *P1* est situé aussi bien à l'interface entre les commutateurs (*NNI*) qu'entre le client et le premier commutateur de raccordement (*UNI*). Un protocole unique facilite la maintenance et la gestion pour l'opérateur.
2. Le protocole entre clients est orienté connexion puisqu'il y a un premier paquet d'établissement de la communication.
3. Le numéro de référence sert à identifier les données qui circulent sur la communication (seul le premier paquet contient les adresses complètes des clients, les suivants se contentant de ce numéro de référence). Par ailleurs, un client peut gérer plusieurs dialogues simultanément, le numéro de référence diffère d'un dialogue à l'autre. C'est l'utilisateur qui choisit le numéro au moment de l'établissement du dialogue.
4. Les commutateurs enregistrent la trace de la communication afin de rechercher le chemin une seule fois, sur le premier paquet. Les autres paquets suivent alors le même chemin, le commutateur allant lire dans sa table comment les aiguiller. Le traitement des paquets de données en est donc accéléré.
5. Le service offert est de grande qualité, garanti par les fonctions de contrôle d'erreurs, de flux et de séquencement : toutes les données traversent le réseau, sur le même chemin et arrivent dans l'ordre.

6. Le nouveau service peut être de même qualité que le précédent tout en offrant un meilleur temps de traversée du réseau, si la transmission à l'intérieur du réseau est de très bonne qualité : en effet, inutile de mettre en œuvre un contrôle d'erreurs qui ralentirait les échanges s'il n'y a pas d'erreurs… Il est intéressant dans ce cas pour l'opérateur d'alléger son protocole.
7. Le service est inévitablement moins bon, car rien ne garantit la qualité de transmission entre les clients et le premier commutateur de raccordement.
8. Ce dernier service, à la différence de tous les précédents, n'est plus orienté connexion. Aucun contrôle n'est effectué dans le réseau : toutes les fonctions de contrôle sont donc reportées sur les utilisateurs eux-mêmes, qui devront anticiper les carences du réseau.

Remarque

Le réseau décrit au début de cet exercice illustre le fonctionnement du réseau Transpac, proposé en France dans les années 1980. Transpac a connu un très grand succès, en particulier parce qu'il transportait les données échangées entre les serveurs minitel et leurs clients. Transpac utilisait X.25, le protocole orienté connexion normalisé par l'ITU, aussi bien comme *UNI* que comme *NNI*. La norme X.25 définit des niveaux Liaison et Réseau fiables. L'inconvénient d'une telle architecture est sa lourdeur, donc sa lenteur d'exécution.
Face à la demande insistante des clients pour des débits plus élevés, l'opérateur doit se tourner vers des protocoles allégés ; Transpac n'a pas échappé à cette évolution. Le protocole IP a ensuite tout balayé sur son chemin dans les années 1990, et les technologies sans connexion d'Internet l'ont supplanté. L'offre X.25 prendra fin en 2011.

Exercice 9 : délais d'exécution d'un service

Solution

1. *A* utilise la seule primitive qui existe : *REQUÊTE_ENVOI*. Si le prestataire n'est pas opérationnel, si le réseau utilisé n'est pas disponible ou est chargé, si le destinataire n'est pas en mesure de recevoir, le service n'est pas exécuté ou est exécuté avec lenteur. L'utilisateur *A* n'est généralement pas informé des multiples raisons d'un échec dans l'exécution du service ou d'un dysfonctionnement. Il doit convenir de règles particulières spécifiques à lui et à *B* pour gérer les éventuelles défaillances du service.

   Du côté du récepteur, une seule primitive signale l'arrivée d'un message *INDICATION_RÉCEPTION (adresse_émetteur, données)*. *B* n'a aucun moyen de savoir s'il y a eu d'autres données auparavant, ni même si celles qu'il reçoit sont les plus récentes émises par *A* ! Notons enfin que le message transmis par le prestataire doit contenir les adresses complètes de l'émetteur et du destinataire. Si *A* doit envoyer un fichier découpé en plusieurs morceaux (le taux d'erreur en ligne impose ce découpage [voir exercice 1]), c'est à lui d'anticiper pour que *B* dispose des informations nécessaires à la vérification de la bonne réception des différents morceaux et la gestion de leur réassemblage.

2. Le second service est beaucoup plus riche. *A* doit établir une connexion avec *B* par la primitive *REQUÊTE_CONNEXION (adresse_B, référence, paramètres d'échanges proposés)*. On supposera, dans cet exemple, que le paramètre *référence* vaut 0 et que, au bout d'un certain temps, *A* reçoit *CONFIRM_CONNEXION (adresse_B, 0, paramètres d'échanges acceptés)*. Cela signifie que *B* a été sollicité et a répondu favorablement. La communication étant établie, *A* peut demander l'expédition de son message par *REQUÊTE_ENVOI (0, données)* dans lequel il n'a plus besoin de préciser l'adresse de *B*, la référence 0 suffit. Une fois qu'il a reçu *SERVICE_EXÉCUTÉ (0)*, il sait que son message a bien été remis à *B* et peut demander la fermeture de la connexion par une *REQUÊTE_FERMETURE (0, fin normale)*. *B* sera informé de la demande de fermeture de la connexion (ici normale, à l'initiative de *A*). Ce fonctionnement est évidemment lourd pour un seul message.
3. Dans le cas du schéma de la figure 3.21 du livre, la connexion a été ouverte pour transférer plusieurs messages et le protocole du prestataire détecte les erreurs de transmission et les corrige par retransmission. Le délai d'exécution de chaque service *REQUÊTE_ENVOI* est donc variable. Les performances sont évidemment meilleures avec anticipation : *A* n'attend pas de savoir que sa première primitive a été correctement exécutée pour demander l'exécution d'une seconde. Il est alors nécessaire que la réponse explicite à quelle demande elle correspond (ce qui n'est pas mentionné sur la figure).

   Les trois premières demandes sont exécutées normalement : délai = 2,5 *t*.

   Le quatrième message est erroné, le protocole réagit à la réception du cinquième en demandant la retransmission de tous les messages depuis le quatrième (réponse `ret 3`). Les messages s'accumulent à l'interface *A* – prestataire de service. Les 10 demandes de *A* sont déjà déposées alors qu'arrive la confirmation de l'exécution du quatrième dont le délai a été de 5 *t* (deux fois plus que normalement). On voit ensuite que tout se passe bien dans la transmission, et donc que les délais se raccourcissent progressivement : 4,5 *t* puis 4 *t*, 3,5 *t*, 3 *t* et enfin 2,5 *t*, retour à une situation normale. On suppose dans ce cas que le prestataire dispose de mémoires tampon de taille suffisante pour absorber les demandes de *A*.

Remarques

1. Le protocole du prestataire de service numérote les données modulo 8 dans cet exemple. Les numéros portés par les messages sont de fait indépendants de ceux que A et B pourraient utiliser entre eux.
2. Alors que le rythme d'émission de A était régulier, on constate que, du côté du récepteur, les réceptions font apparaître un grand intervalle de silence puis, d'un coup, une rafale de données : les fonctions de contrôle de flux et de contrôle de congestion sont fondamentales pour limiter de telles accumulations. Ces fonctions peuvent être mises en œuvre à l'interface entre clients et prestataire comme au sein du réseau du prestataire lui-même.

Chapitre 4

Les réseaux locaux d'entreprise

Pour répondre à leurs besoins en informatique distribuée, les entreprises ont mis en œuvre des *réseaux locaux d'entreprise*, constitués d'un ou de plusieurs réseaux locaux ou LAN (*Local Area Network*), qui utilisent des protocoles simples car les distances couvertes sont courtes (de quelques centaines de mètres à quelques kilomètres) et les débits importants (jusqu'à plusieurs gigabits par seconde). Nous détaillons les différentes techniques d'accès au support, spécifiques de ce type de réseau, puis nous analysons le fonctionnement des réseaux locaux de première génération pour mieux comprendre leurs évolutions technologiques. Enfin, nous abordons les réseaux sans fil.

Problèmes et exercices

Exercice 1 : câbler un petit réseau local à la maison

Solution

1. Il faut tout d'abord disposer des matériels et des logiciels appropriés. Pour ce faire, vous devez choisir le réseau local que vous voulez créer (Ethernet ou réseau sans fil) et la topologie physique que vous allez utiliser. Vous optez pour des cartes Ethernet, afin de créer un réseau simple et peu coûteux. Vous devez ensuite décider comment raccorder vos ordinateurs : topologie physique en bus ou en étoile ?

   La topologie en bus est la solution la plus économique si vos ordinateurs sont situés dans la même pièce. La topologie en étoile, désormais la plus populaire, impose l'achat d'un concentrateur dont le prix dépend du nombre de ports disponibles. Cette dernière solution vous permet de faire évoluer plus aisément votre installation (mais aurez-vous plus d'une dizaine de machines à la maison ?).

   Vous décidez donc de raccorder vos machines en bus. Les étapes de votre installation sont : achat et assemblage des différents matériels, installation des logiciels, configuration des adresses IP.

   Au terme de la première étape, vous avez installé les matériels suivants :

- un câble Ethernet torsadé ;
- autant de prises RJ45 que vous raccordez d'ordinateurs ;
- un concentrateur à huit ports ;
- des cartes réseau (ou cartes Ethernet), une par ordinateur à connecter. Choisissez plutôt des cartes équipées de connecteurs RJ45.

   Vous devez également disposer, sur chaque machine connectée, des logiciels de communication :

- un pilote (*driver*) pour chaque carte réseau, en général fourni par le constructeur de la carte ;
- une pile TCP/IP par ordinateur, souvent fournie avec le système d'exploitation ;
- un navigateur par ordinateur si vous vous voulez surfer sur Internet et si vous avez souscrit un abonnement auprès d'un fournisseur d'accès.

   Il vous reste à tout assembler pour achever la deuxième étape ! Pour la troisième, les systèmes d'exploitation modernes possèdent souvent des fonctions de type *Plug and Play* (littéralement : branchez et jouez) ; les pilotes et autres logiciels sont alors très faciles à installer. Reste la dernière étape : l'affectation des adresses IP à toutes les machines. Nous verrons cette étape au chapitre 6, qui traite du protocole IP.

2. La conséquence immédiate de ce choix est que toute votre belle installation est à jeter ! Si vous souhaitez installer le réseau sans fil le plus simple qui soit, vous équipez tous les ordinateurs avec une carte Wi-Fi au lieu de la carte réseau précédente. Toutes

les applications (partage de l'imprimante, jeux en réseau…) qui utilisent la pile TCP/IP seront utilisables sur vos machines. Cette architecture est une architecture *ad hoc*, décrite dans le standard 802.11.

Exercice 2 : bouchon de terminaison et période de vulnérabilité

Solution

1. Aucune transmission n'est possible. Le bouchon a un rôle électrique ; son impédance doit être bien adaptée de telle sorte que les signaux ne soient pas réfléchis en arrivant aux extrémités du câble. La réflexion est une source de bruit qui perturbe toutes les transmissions.
2. Si les stations sont réparties tous les 15 m, la distance entre les deux stations les plus éloignées l'une de l'autre est de $15 \times 7 = 105$ m. La période de vulnérabilité correspond au temps de propagation aller et retour entre les deux stations les plus éloignées, soit :

$$2 \times 105/250 = 0{,}84\ \mu s$$

Remarque

Sur un bus aussi court, la probabilité d'une collision est très faible : il faudrait que deux équipements (ou plus) aient écouté et pris la décision d'émettre dans le même intervalle de 0,84 µs, d'où l'intérêt d'utiliser des bus plutôt courts.

3. Soit D la distance maximale. Si la période de vulnérabilité vaut 51,2 µs, alors D est égal à :

$$D = \text{période de vulnérabilité} \times \text{vitesse} = 51{,}2 \times 250 = 1\,280 \text{ m}$$

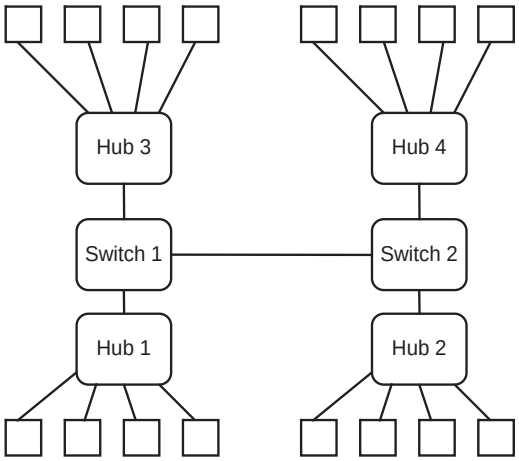
Exercice 3 : notion de domaine de collision

Solution

1. Il n'existe qu'un seul domaine de collision, puisque chaque concentrateur propage les données qui circulent sur ses ports à tous les autres concentrateurs. Pour une charge de réseau importante, les collisions risquent d'être trop nombreuses pour un service satisfaisant.
2. Il y a maintenant quatre domaines de collision (un par concentrateur). En effet, les commutateurs stockent les trames avant de les réémettre sur un autre port, ce qui minimise les collisions.

3. Pour éviter toute collision, on élimine les concentrateurs et on raccorde chaque station au port d'un commutateur. Vu la taille du réseau, un seul commutateur suffit pour raccorder tous les équipements.

Figure 4.1
Réseau de l'entreprise avec les deux commutateurs.



Remarque

Le réseau proposé peut à la rigueur s'envisager pour un particulier, mais il est peu réaliste dans une entreprise ! L'exemple ne sert qu'à mettre en évidence le partitionnement d'un réseau pour minimiser les collisions, et donc améliorer la qualité du service. Nous verrons des exemples plus réalistes dans les exercices du chapitre 7.

Exercice 4 : adresse MAC

Solution

L'adresse MAC est l'adresse physique de la carte Ethernet. C'est le numéro de série de cette carte, défini par le constructeur de la carte. Les constructeurs ont des préfixes uniques au monde (3 octets) et numérotent ensuite leurs cartes sur les 3 octets suivants : deux cartes ne peuvent jamais avoir le même numéro de série. Il est donc impossible qu'un autre ordinateur possède la même adresse.

Remarque

Il est aujourd'hui possible de flasher la PROM qui contient l'adresse MAC. Bien que cette technique viole la règle d'unicité des adresses MAC au sein d'un réseau donné, elle évite la mise à jour des tables de correspondance entre adresses MAC et adresses IP en cas de remplacement d'une carte réseau défectueuse, par exemple.

Exercice 5 : taille minimale des trames Ethernet

Solution

1. Pour que toutes les stations détectent la collision, il faut qu'on ait $T = p + \Delta$, qu'on peut borner supérieurement par $T = 2\,p$.
2. Puisque $T = M/8\Delta$, on trouve $M = 16\,p\Delta$, en remplaçant T par sa valeur dans l'expression ci-dessus.
3. À 10 Mbit/s, chaque bit dure 0,1 µs. 51,2 µs correspond au temps d'émission de 512 bits (64 octets).
4. Les répéteurs introduisent un délai supplémentaire, ils interviennent donc dans la valeur de p.

Remarque

On comprend pourquoi la norme 802.3 impose une taille minimale pour les messages émis par les équipements d'un réseau local de type CSMA/CD. Les récepteurs procèdent ensuite au tri entre les « résidus de collision » trop courts et les « vraies » trames d'une longueur suffisante.

Exercice 6 : débit utile

Solution

1. Le débit utile maximal s'obtient de manière théorique si une station unique émet en permanence (en respectant l'espace entre trames) des trames de longueur maximale. On obtient alors :

   longueur totale équivalente d'une trame en octets = 8 (préambule) + 6 (adresse destinataire) + 6 (adresse émetteur) + 2 (longueur ou type) + 1 500 (contenu utile) + 4 (bloc de contrôle d'erreurs) + 12 (correspondant au silence entre trames) = 1 528 octets.

   Le débit utile vaut : $10 \times (1\,500/1\,528) = 9{,}82$ Mbit/s, soit un rendement de 98,2 %.

   Il s'agit bien évidemment d'un calcul théorique : il est impossible d'atteindre un tel rendement dans la pratique, dès que plusieurs équipements tentent d'émettre. Il y aura des silences et des collisions qui entraîneront d'éventuels silences et/ou collisions supplémentaires.

Remarque

En pratique, on considère qu'un rendement de 50 à 60 % est une valeur limite. Si le trafic devait être plus important, les performances s'effondreraient. Cet exercice montre l'intérêt des commutateurs pour segmenter les réseaux locaux.

2. Le débit est de 10 Mbit/s, 1 bit dure $1/(10 \times 10^6) = 0{,}1$ µs soit, avec la vitesse de propagation de 200 m/µs, un temps correspondant au parcours dans 20 m de câble. Dans le réseau local dont la longueur est 800 m, cela suppose qu'il y a, à un instant donné, 800/20 = 40 bits.
3. Les données sont trop courtes, la trame est complétée avec du bourrage pour atteindre 46 octets.

   Longueur totale équivalente d'une trame en octets = 8 (préambule) + 6 (adresse destinataire) + 6 (adresse émetteur) + 2 (longueur ou type) + 46 (contenu utile) + 4 (bloc de contrôle d'erreurs) + 12 (correspondant au silence entre trames) = 84 octets.

   Le débit utile réel vaut : $10 \times (32/84) = 3{,}81$ Mbit/s, soit un rendement de 38,1 %.

Exercice 7 : simulation de trafic sur Ethernet

Solution

1. Le chronogramme est le suivant :

ST	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
	A	*A*	*A*	*A*	*A*	*A*	*X*	*B*	*B*	*B*	*B*	*B*	*B*	*X*		*X*		*D*	*D*	*D*	*D*	*D*	*D*	*C*	*C*	*C*	*C*	*C*	*C*

   Commentaire : à la date 0, *A* démarre, le support est libre et sa trame dure 6 ST, donc de 0 à 5 ST. À $t = 2$ ST, *B* et *C* veulent transmettre mais le support est occupé : elles attendent. À $t = 5$ ST, *D* veut transmettre, le support est occupé, donc elle attend.

   À $t = 6$ ST, le support devient libre, toutes les stations en attente (*B*, *C* et *D*) tentent leur chance : il y a collision. *B*, *C* et *D* suspendent leur transmission et démarrent une attente aléatoire. Celle-ci sera nulle pour *B* et de 1 ST pour les deux autres. À $t = 7$ ST, *B* tente sa chance une nouvelle fois. Le support est libre, sa trame dure 6 ST, elle va de 7 à 12 ST.

   À $t = 8$ ST, *C* et *D* veulent faire une nouvelle tentative. Le support étant occupé, elles attendent.

   À $t = 13$ ST, le support devient libre. Toutes les stations en attente (*C* et *D*) tentent leur chance : il y a une nouvelle collision. *C* et *D* suspendent leur transmission et démarrent une deuxième attente aléatoire, valant 1 ST pour chacune, conformément au tableau précédent.

   À $t = 14$ ST, il y a un silence, car les deux stations *C* et *D* attendent la fin du délai aléatoire et, à $t = 15$ ST, elles tentent leur chance, une nouvelle fois ensemble ! Il y a de nouveau collision. Cette fois, le délai aléatoire est heureusement différent pour les deux stations qui vont donc réussir à transmettre : pour *D*, à $t = 17$ ST ; pour *C*, à $t = 23$ ST puisque, à sa troisième tentative (à $t = 16 + 5 = 21$ ST), le support est occupé par *D*.
2. Le taux d'utilisation du canal est de 24/29, soit de 82 %.

Exercice 8 : risque de collisions et délai moyen d'attente

Solution

1. *A* n'a subi qu'une collision, donc le délai aléatoire qu'il a tiré au sort est 0 ou 1 fois l'intervalle ST. *B* en a subi deux successives, donc le délai qu'il a pu tirer au sort est uniformément réparti entre 0 ST, 1 ST, 2 ST et 3 ST.

   Soit p la probabilité d'une nouvelle collision. Pour qu'un tel événement se produise, il faut que les deux équipements aient tiré au sort simultanément 0 ou simultanément 1. Notons *NA* (respectivement *NB*) la durée du délai pour *A* (respectivement *B*). On obtient :

   $$p = \text{Proba}\,[NA = 0] \times \text{Proba}\,[NB = 0] + \text{Proba}\,[NA = 1] \times \text{Proba}\,[NB = 1]$$

   $$p = 1/2 \times 1/4 + 1/2 \times 1/4 = 1/4 = 0{,}25$$

2. Si *B* a déjà subi cinq collisions, le délai qu'il va tirer est réparti entre 0 ST et 31 ST.

   $$p = \text{Proba}\,[NA = 0] \times \text{Proba}\,[NB = 0] + \text{Proba}\,[NA = 1] \times \text{Proba}\,[NB = 1]$$

   $$p = 1/2 \times 1/32 + 1/2 \times 1/32 = 1/32$$

3. Le nombre de collisions déjà subies par un équipement détermine la taille de l'intervalle dans lequel il tire au sort son délai d'attente. Le temps moyen d'attente avant retransmission pour un essai donné est en effet égal à la moitié de l'intervalle de tirage, puisqu'il s'agit d'une loi uniforme. Le temps moyen cumulé pour n tentatives est donc la somme de chaque temps moyen, pour n allant de 1 à 16.

   Soit ST la durée du Slot-Time.

   Si $n = 0$, l'équipement n'a subi aucune collision et $T0 = 0$.

   Dans le cas où l'équipement a subi n collisions au total, avec n inférieur ou égal à 10, avant de réussir sa transmission, son délai d'attente se calcule comme suit :

   Le délai d'attente a une valeur nulle avant la première transmission. Après la première collision, comme N vaut 0 ou 1, on a $D1 = (0 + 1)/2 \times \text{ST} = \text{ST}/2$. Après la deuxième collision, N vaut 0, 1, 2 ou 3, donc on a $D2 = (0 + 1 + 2 + 3)/4 \times \text{ST} = 3\ \text{ST}/2$. Après la troisième collision, N vaut 0, 1, 2, 3, 4, 5, 6 ou 7. On obtient : $D3 = (0 + 1 + 2 + 3 + 4 + 5 + 6 + 7)/8 \times \text{ST} = 7\ \text{ST}/2$ et ainsi de suite jusqu'à n. Donc :

   $$Tn = D0 + D1 + D2 + \ldots + Dn = 0 + \text{ST}/2 + 3\ \text{ST}/2 + \ldots + (2^n - 1)\ \text{ST}/2$$
   $$= (2^n - (n + 1)/2) \times \text{ST}$$

   Si l'équipement a subi n collisions au total, avec n compris entre 11 et 15 (bornes incluses), le calcul est légèrement différent du précédent puisque : $D10 = D11 = D12 = D13 = D14 = D15$. On trouve alors :

   $$Tn = T10 + (n - 10) \times D10$$

Exercice 9 : étude de spécifications au niveau MAC

Solution

1. Le temps d'émission d'une trame de 4 500 octets correspond au temps de propagation de 1 bit plus le temps de propagation de la trame (durée de 1 bit × nombre de bits), d'où :

   $$t = 4\,500 \times 8/100 \times 106 + 10 - 3 = 360\ \mu s + 1000\ \mu s = 1\,360\ \mu s\ (1{,}36\ ms)$$

2. La durée d'émission de la plus petite trame correspond à la période de vulnérabilité, c'est-à-dire à deux fois le temps de propagation entre les stations les plus éloignées, soit :

   $$T = 2 \times 10^{-3}/100 \times 10^{6} = 2\ ms$$

3. La trame minimale devrait durer 2 ms, donc sa longueur devrait être de :

   $$2 \times 10^{-3} \times 100 \times 10^{6} = 200 \times 10^{3}\ \text{bits, soit } 25\,000\ \text{octets}$$

4. Le rapport ρ vaut :

   $$\rho = 4\,500/25\,000 = 0{,}18, \text{ soit un rendement de } 18\ \%$$

5. Le débit réel est de : $100 \times 10^{6} \times 0{,}18 = 18$ Mbit/s.

6. Non, à cause de la taille démesurée du champ de remplissage des trames (au minimum 20 500 octets). La période de vulnérabilité est trop importante. CSMA/CD est un mauvais choix dans ce cas.

Remarque

On comprend mieux pourquoi il a fallu modifier la méthode d'accès pour les réseaux Gigabit Ethernet et 10 Gigabit Ethernet.

Exercice 10 : Ethernet commuté

Solution

1. La topologie physique est en étoile, le débit de 100 Mbit/s sur paires métalliques.
2. Avec un concentrateur, lorsqu'un équipement émet vers un autre, tous les équipements du réseau reçoivent l'information. Le débit de 100 Mbit/s est partagé entre les utilisateurs, et les transferts de données se font à l'alternat. Un concentrateur est un équipement très bon marché.

   Avec un commutateur, si un équipement émet vers un autre, seul le destinataire reçoit l'information. Chaque utilisateur emploie un débit de 100 Mbit/s et les transferts de données sont bidirectionnels simultanés. Un commutateur est plus onéreux, mais le rapport prix/performances justifie le supplément.

3. Si le commutateur a une capacité suffisante, chaque équipement, directement relié au commutateur, peut disposer d'un débit théorique dédié de 100 Mbit/s dans les deux sens de transmission. Puisque les cinq équipements communiquent avec le même serveur, le lien entre le serveur et le commutateur est en fait partagé entre les cinq communications : un débit maximal de 20 Mbit/s est offert à chaque dialogue.

Exercice 11 : Gigabit Ethernet

Solution

1. Le temps d'émission d'une trame de 512 octets est $512 \times 8/10^9$, soit environ 4 ms.
2. On peut en déduire que la période de vulnérabilité est au plus égale à 4 ms.

Remarque

Cet exercice complète le travail fait à l'exercice 10 et illustre un autre aspect de la nécessité d'adapter les méthodes d'accès aux contraintes liées au bon fonctionnement d'un réseau.

Exercice 12 : transmissions dans les réseaux sans fil

Solution

1. Une station ne peut pas émettre et recevoir en même temps sur la même fréquence, car il en résulte un phénomène d'« éblouissement » de l'antenne de réception. Une fréquence est donc nécessaire pour l'émission et une autre pour la réception.
2. Le récepteur ne peut pas s'assurer que le signal reçu n'a pas été perturbé par celui d'une autre station, puisque les signaux émis et reçus se trouvent dans des fréquences différentes. La détection de collision telle qu'elle est pratiquée dans Ethernet est donc inapplicable.
3. Le moyen le plus simple consiste à détecter une éventuelle émission pendant un temps supérieur au délai de propagation le plus long, en tenant compte de la portée du réseau.
4. La technique utilisée minimise les risques de collisions pour deux raisons principales : d'une part, chaque station de la zone concernée peut estimer le temps occupé par les autres stations, grâce au mécanisme de réservation. D'autre part, les trames de contrôle RTS et CTS sont courtes, donc la probabilité de collision est faible. Pour autant, les collisions ne peuvent être totalement éliminées puisque la simple écoute du support ne suffit pas pour les détecter (cas des stations cachées).

Exercice 13 : performances des WLAN

Solution

1. Globalement, on peut considérer que le débit disponible sera divisé par le nombre de stations actives dans le réseau sans fil. Pour obtenir un débit suffisant, il faut accroître le nombre de cellules.
2. La multiplication des cellules pour couvrir une aire géographique offre un avantage et un inconvénient : lorsque la portée d'une cellule est diminuée, la puissance nécessaire à l'émission est plus faible, donc la durée de vie des batteries est préservée. Par ailleurs, l'augmentation du nombre de cellules impose d'utiliser plusieurs fréquences distinctes. L'affectation des fréquences risque d'être compliquée à gérer si l'on veut éviter les interférences au sein d'une même cellule ou entre cellules voisines. Par ailleurs, les différences de réglementation entre pays viennent complexifier le problème.
3. D'après le théorème de Shannon, la relation entre le débit binaire et la bande passante donne un débit maximal de 500 kbit/s. Si l'on change de canal toutes les 400 ms, on transmet environ 1 Mbit/s. On peut atteindre 2 Mbit/s en divisant par deux le temps imparti à l'utilisation de chaque canal. Dans la technique DSSS, en utilisant le même théorème, on obtient un débit maximal de 11 Mbit/s.
4. La bande de 2,4 GHz occupe une bande de fréquences de 85 MHz. D'après nos hypothèses, si toutes les bandes étaient placées les unes derrière les autres, le canal 14, centré sur la fréquence 2,763 GHz, occuperait la bande comprise entre 2,752 GHz et 2,774 GHz. Il faudrait occuper une bande de fréquences de 373 MHz. Les bandes allouées aux canaux se chevauchent donc pour tenir dans la bande allouée. Pour éviter les interférences dans une cellule ou avec les cellules voisines, il faut choisir des bandes de fréquences suffisamment espacées les unes des autres, par exemple des bandes espacées de 25 MHz l'une de l'autre. La figure 4.2 montre comment utiliser les bandes 1, 6 et 11.

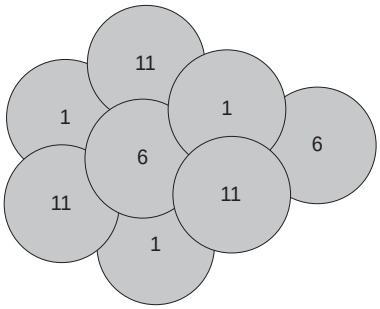


Figure 4.2
Utilisation des bandes 1, 6 et 11 dans une cellule.

Exercice 14 : performances dans les WPAN

Solution

1. Puisqu'un slot dure 625 µs, le temps est découpé en 1 600 slots par seconde. Un esclave ne peut utiliser que 800 trames par seconde.
2. Une trame Bluetooth dure au maximum : 5×652 µs = 3 125 µs.
3. Puisque Bluetooth et Wi-Fi opèrent dans la même bande de fréquences et utilisent les mêmes canaux, ils interfèrent. Il faut que les groupes de travail 802.11 et 802.15 se concertent pour proposer une solution pour la coexistence de ces deux réseaux dans un même local. Dans le cas contraire, les entreprises devront choisir une des deux solutions en compétition. Il en résultera un rapport de force qui pourra contraindre l'un des deux protagonistes à modifier son standard pour supprimer les interférences avec l'autre.
4. Avec un débit de 64 kbit/s, la durée de 1 bit vaut : 15,625 µs. On peut transmettre 200 bits dans une trame de longueur maximale.
5. Le moyen le plus simple pour minimiser les collisions consiste à détecter une éventuelle émission pendant un temps supérieur au délai de propagation le plus long, en tenant compte de la portée du réseau.

Exercice 15 : architecture d'un réseau sans fil plus étendu

Solution

1. Vous commencez par vous documenter sur les réseaux sans fil. Plusieurs standards coexistent : 802.11a, 802.11b et 802.11g. Les deux premiers sont incompatibles entre eux ; le dernier offre le même débit que 802.11a tout en étant compatible avec le standard 802.11b.
2. Vous décidez donc d'acquérir des matériels certifiés Wi-Fi, supportant le standard 802.11f[2], de façon à installer votre WLAN à partir d'équipements de différents constructeurs. Vous choisissez le standard 802.11g, qui vous offre des débits allant jusqu'à 54 Mbit/s, sans les contraintes strictes imposées aux WLAN fonctionnant en 802.11a. Votre réseau sera un réseau *à infrastructure*, vous permettant d'installer suffisamment de bornes d'accès pour bien desservir l'ensemble du bâtiment.
3. Au minimum, il faut une base (on parle aussi de *borne* ou de *point d'accès*) et autant de cartes Wi-Fi que d'ordinateurs portables souhaitant utiliser le WLAN. La base doit posséder la fonction de pont, afin de la raccorder au réseau Ethernet filaire de l'entreprise. En outre, vous devez acheter des antennes de différents types, selon l'utilisation qui en sera faite et le type de local à raccorder. Le tableau 4.1 récapitule les différents types d'antennes et l'usage auquel elles sont destinées.

2. Standard autorisant le roaming (déplacement dans des zones couvertes par plusieurs bases) entre des bases provenant de constructeurs différents.

Tableau 4.1 : Les différents types d'antennes pour réseaux sans fil

Type	Locaux à desservir
Verticale	Salles de réunion, bureaux
Dipôle	Couloirs (zones étroites et longues)
Sectorielle	Salles de réunion ou halls d'entrée
Yagi	Liaison entre bâtiments proches
Parabolique	Liaison entre immeubles éloignés

4. Les différentes bornes d'un WLAN étendu doivent se situer dans le même domaine de collision et appartenir au même sous-réseau IP, afin que vos utilisateurs se déplacent dans le bâtiment sans perdre la connexion au réseau (à cet effet, vous avez choisi des matériels compatibles supportant 802.11f).

   Pour desservir l'immeuble, vous devez placer vos bornes de sorte que leurs zones de couverture se chevauchent. Pour connaître le nombre de bases nécessaires, vous pouvez utiliser des outils qui simulent votre futur réseau. Plus simplement, vous vérifiez votre installation en vous promenant avec un portable (doté d'un logiciel de test affichant la puissance du signal reçu). Vous vérifiez l'absence de zones d'ombre et vous vous assurez que la puissance du signal reçu est suffisante dans toutes les zones[3].

5. Deux possibilités s'offrent à vous, selon la distance entre les deux bâtiments : soit vous augmentez la portée du WLAN en exploitant le mode répéteur de certaines bornes, à l'aide d'antennes de type *parabolique* ou *yagi*, soit vous installez dans l'autre bâtiment de nouvelles bornes, raccordées comme les autres par un Ethernet filaire et obéissant aux mêmes conditions (même domaine de collision et même sous-réseau IP).

   Avec la mise en cascade de certaines bornes en mode répéteur, vous divisez le débit utile par deux, puisque les bornes se partagent le même canal pour communiquer entre elles et avec les stations (la borne faisant office de répéteur transmet la trame à l'autre borne avec la même fréquence). Cette solution ne sera pas retenue si vous souhaitez disposer du débit le plus élevé possible. Dans ce cas, les contraintes d'appartenance peuvent être délicates à respecter selon la topologie des lieux.

6. La configuration de votre WLAN se fait en entrant les paramètres système et les paramètres de communication des postes clients et des différentes bases. Parmi les paramètres système figurent : le nom du WLAN (SSID ou ESS selon le type de réseau sans fil), le mode de fonctionnement de l'interface sans fil (en mode constamment actif ou en mode veille pour économiser l'énergie), le type de réseau (réseau à infrastructure). Il faut également choisir les paramètres de communication : le débit (le plus élevé possible ou à spécifier manuellement), les canaux de communication, la taille maximale d'une trame, la puissance d'émission. Les différents champs doivent être remplis en veillant à ce que tous les paramètres soient identiques dans tous les composants du WLAN.

Remarque

Nous n'avons pas abordé dans cet exercice les aspects liés à la sécurité de fonctionnement du réseau sans fil. Nous les verrons à l'exercice 12 du chapitre 10.

3. Certains équipements proposent une aide pour réaliser ces tests (mode *survey*).

Chapitre 5

Le protocole IP (*Internet Protocol*)

Le protocole IP transfère les données à travers une interconnexion de réseaux. Il cherche un chemin pour véhiculer les paquets (*datagrammes*) d'un émetteur à un destinataire, chacun étant identifié par son adresse IP ; IP est utilisé par les protocoles de la couche Transport : TCP et UDP. Le module IP ne fournit aucune garantie d'acheminement correct des paquets et ignore le comportement du module IP des autres machines. Chaque paquet est traité indépendamment des autres : cela signifie que ceux-ci peuvent être mélangés, dupliqués, perdus ou altérés ! Pour comprendre le fonctionnement du protocole IP, nous étudions les adresses IP elles-mêmes ainsi que leur correspondance avec les adresses MAC, puis le traitement effectué par un module IP et le format du paquet. Nous évoquons enfin brièvement les particularités de la version 6 du protocole.

Problèmes et exercices

Exercice 1 : principes généraux de l'adressage

Solution

1. Séparer l'adresse en deux parties réduit la taille mémoire des routeurs, qui ne conservent que l'adresse des (sous-)réseaux et celles des stations des (sous-)réseaux directement rattachées. En effet, la séparation entre l'adresse du réseau et celle de la station attachée au réseau permet un routage effectif dans les routeurs uniquement d'après l'adresse du réseau. L'adresse complète n'est utilisée qu'une fois le paquet arrivé dans le routeur connecté au réseau destinataire.
2. L'adresse IP doit non seulement être unique, mais aussi refléter la structure de l'interconnexion. La partie réseau de l'adresse dépend donc du réseau auquel est connectée la station : toutes les machines connectées au même réseau physique ont le même préfixe réseau.

Exercice 2 : informations de configuration

Solution

1. *A* est dans le réseau `143.27.0.0`, dans le sous-réseau `143.27.64.0` (on obtient `64` en faisant le ET entre les nombres `102` et `192` écrits sur 8 bits, soit `01100110` ET `11000000`. Le résultat donne : `01000000` = `64`). Il y a donc 2 bits pour définir les sous-réseaux. L'adresse de diffusion dans ce sous-réseau est `143.27.127.255` (on obtient `127.255` en remplaçant les 14 bits prévus pour l'identifiant de machine par des `1`).
2. *B* est dans le réseau `143.27.0.0`, mais pas dans le même sous-réseau (il est dans le sous-réseau `143.27.128.0`). Il ne peut donc pas utiliser la même adresse de routeur par défaut (le routeur par défaut est obligatoirement dans le sous-réseau de l'utilisateur).

Remarque

Ce réseau ne comprend qu'un routeur possédant deux interfaces internes et une interface vers le monde extérieur. *A* et *B* utilisent le même routeur pour transmettre des messages entre eux ou vers l'extérieur. Chaque utilisateur désigne le routeur par l'adresse IP de l'interface réseau qu'il connaît. On voit donc bien que l'adresse IP ne définit pas une machine mais une interface réseau.

Exercice 3 : correspondance adresse MAC/adresse IP

Solution

1. La machine *1* doit rechercher l'adresse MAC de la machine *2* qu'elle connaît par son adresse IP_2. Elle consulte sa table ARP. Si celle-ci contient l'information, le problème est résolu. Dans le cas contraire, la machine *1* utilise le protocole ARP en diffusant une trame qui contient la requête suivante : « Je cherche l'adresse MAC de la machine dont je connais l'adresse IP_2. » La trame étant diffusée, tous les équipements du réseau local la reçoivent. Seul celui qui est concerné par l'échange, c'est-à-dire ici la machine *2*, répond par une trame contenant la réponse ARP suivante : « Je suis la machine IP_2, mon adresse MAC est PH_2. » En recevant cette réponse, la machine *1* met à jour sa table ARP en ajoutant une ligne : « IP_2 correspond à PH_2. »
2. Si la machine *2* est sur un autre réseau local, elle possède une adresse IP_2 qui n'appartient pas au même réseau qu'IP_1 (la machine *1* le sait en utilisant son masque de sous-réseau). Le paquet doit être acheminé à l'extérieur du réseau ; il est envoyé au routeur. L'adresse IP_R du routeur est présente dans le fichier de configuration de la machine *1*. Dans le cas où elle ignore l'adresse MAC PH_R du routeur, il lui faut rechercher cette adresse au moyen d'une requête ARP comme à la question 1. Puis la machine *1* émet dans le réseau local une trame dont les adresses physiques sont : destinataire = PH_R et émetteur = PH_1. Cette trame transporte un paquet IP dont les adresses logiques sont : émetteur = IP_1 et destinataire = IP_2.

Exercice 4 : sous-réseaux

Solution

Tableau 5.1 : Adresses IP, classes et adresses particulières

Adresse IP	124.23.12.71	124.12.23.71	194.12.23.71
Masque de sous-réseau	255.0.0.0	255.255.255.0	255.255.255.240
Classe	*A*	*A*	*C*
Adresse du réseau auquel appartient la machine	124.0.0.0	124.0.0.0	194.12.23.0
Adresse de diffusion dans le réseau	124.255.255.255	124.255.255.255	194.12.23.255
Adresse du sous-réseau auquel appartient la machine	Pas de sous-réseau	124.12.23.0	194.12.23.64
Adresse de diffusion dans le sous-réseau de la machine		124.12.23.255	194.12.23.79

Exercice 5 : généralités sur le plan d'adressage

Solution

Adresse de classe *B* : `x.y.0.0` avec `x` compris entre `128` et `191`. En l'absence d'hypothèse précise sur le nombre de machines dans chaque sous-réseau et sur l'évolution future du réseau, on considère qu'il suffit de créer deux sous-réseaux (ce qui nécessite 2 bits si on veut éviter les sous-réseaux « plein 0 » et « plein 1 »), donc un masque `255.255.192.0`. Dans les adresses IP des stations, les 16 premiers bits représentent le réseau (`x.y.`), les 2 bits suivants les sous-réseaux (`01` et `10`). Les 14 bits restants désignent la machine elle-même.

Le sous-réseau `01` a pour adresse de sous-réseau `x.y.64.0` ; les adresses des machines vont de `x.y.64.1` à `x.y.127.254` ; l'adresse de diffusion dans ce sous-réseau est `x.y.127.255`. Tout message parvenant au routeur avec une adresse IP dans l'intervalle ci-dessus est diffusé exclusivement dans ce sous-réseau.

Le sous-réseau `10` a pour adresse de sous-réseau `x.y.128.0` ; les adresses des machines vont de `x.y.128.1` à `x.y.191.254` ; l'adresse de diffusion dans ce sous-réseau est `x.y.191.255`. Tout message parvenant au routeur avec une adresse IP dans l'intervalle ci-dessus est diffusé exclusivement dans ce sous-réseau.

Exercice 6 : plan d'adressage particulier

Solution

1. Oui, car une adresse de classe *B* permet d'adresser $2^{16} - 2$ (65 534 machines), soit bien davantage que le nombre de machines installées.
2. Une adresse de classe *C* identifie 254 machines. Il faut 12 adresses de classe *C* pour tous les terminaux.
3. Il faut 4 bits pour identifier 12 sous-réseaux. Le masque vaut donc : `255.255.240.0`.
4. Il reste 12 bits, c'est-à-dire qu'on peut adresser $2^{12} - 2$ machines soit 4 094 machines par sous-réseau.
5. Le sous-réseau n° 1 a pour adresse `139.47.16.0` (les 4 bits de sous-réseau valent `0001`, soit 1 en décimal), donc le sous-réseau n° 9 aura pour adresse réseau : `139.47.144.0` (les 4 bits de sous-réseau valent `1001`, soit 9 en décimal).
6. La machine `7.48` du sous-réseau `139.47.144.0` a pour adresse IP `139.47.151.48`.
7. L'adresse réseau du sous-réseau n° 12 est : `139.47.192.0` ; son adresse de diffusion vaut : `139.47.207.255`.

Exercice 7 : plan d'adressage avec sous-réseaux

Solution

1. L'entreprise dispose de 50 + 7 = 57 machines : une adresse de classe *C* lui suffit. Pour faire apparaître six sous-réseaux (un par groupe et un pour les deux serveurs communs), il faut au moins 3 bits. Il reste alors 5 bits, soit 32 – 2 = 30 adresses disponibles, ce qui convient parfaitement puisque chaque groupe comprend au maximum 11 postes. Le masque de sous-réseau sera `255.255.255.224`. Les cinq groupes d'utilisateurs correspondent aux sous-réseaux `193.22.172.32`, `193.22.172.64`, `193.22.172.96`, `193.22.172.128` et `193.22.172.160`. Les deux serveurs seront dans le dernier sous-réseau `193.22.172.192`.
2. Dans cet exemple, il faut faire quatre sous-réseaux ; on prendra 3 bits pour identifier les sous-réseaux. Les groupes sont de tailles différentes, mais tous comptent au plus 30 postes. 5 bits pour identifier une machine sont suffisants. On pourra utiliser le même masque `255.255.255.224`.

Remarque

Avec CIDR, on pourrait très bien se contenter de 2 bits pour identifier les quatre sous-réseaux, qui seraient alors numérotés de `0` à `3` (on aurait un masque `/26`). Dans ce cas, le réseau global et le sous-réseau 3 auraient la même adresse de diffusion : `193.22.172.255`.

Exercice 8 : CIDR

Solution

L'indication `/22` signifie que les 22 premiers bits sont dévolus à l'adresse réseau et que l'entreprise est libre d'utiliser les 10 bits restants pour ses machines. Elle dispose d'un millier d'adresses, ce qui lui convient.

Le masque de sous-réseau par défaut est alors, en découpant les octets : `11111111 11111111 11111100 00000000`, soit en décimal : `255.255.252.0`.

Exercice 9 : fragmentation des paquets

Solution

1. Le bit MF (*More Fragments*) est à 1 dans tous les fragments sauf le dernier ; le champ Déplacement n'est pas nul, sauf dans le premier fragment, alors qu'un paquet non fragmenté possède un bit MF à 0 *et* un champ Déplacement à 0.

2. Tous les fragments portent le même identifiant (celui du paquet initial). On utilise alors le champ Déplacement pour reconstituer le paquet. Le bit MF est à 0 dans le dernier fragment, à 1 dans tous les autres.
3. Un routeur ne peut pas confondre deux fragments dont les éléments source, destination et place de fragment seraient identiques, car le champ Identifiant du paquet est forcément différent !

Exercice 10 : utilitaire *ping*

Solution

La machine d'adresse IP `193.93.28.7` est opérationnelle puisqu'elle a répondu à la requête d'écho. C'est là le rôle initial de l'utilitaire *ping* : tester si un équipement fonctionne en lui envoyant un message qu'il doit immédiatement renvoyer. Le délai de traversée est très bref, puisqu'on est à l'intérieur d'un réseau local. L'utilitaire *ping* envoie des messages du protocole ICMP (*Echo Request*) qui sont numérotés. Ici, il n'y en a qu'un, donc son numéro de séquence est `icmp_seq` = 0. Le message n'a traversé aucun routeur puisque le champ TTL est à 255, ce qui représente sa valeur maximale.

Remarque

La nouvelle version IPv6 utilisant des adresses au format totalement différent d'IPv4, un nouvel utilitaire baptisé *ping6* a été développé pour proposer les mêmes services que *ping* (voir exercice 11). De même, un nouvel utilitaire baptisé *tracert6* a été développé pour proposer les mêmes services que *traceroute*.

Exercice 11 : utilitaire *ping6*

Solution

La machine d'adresse `IPv6 2001:703:400D:3D15::71:1` est opérationnelle puisqu'elle a répondu à la requête d'écho. Le délai de traversée est moyen et variable. Comme *ping*, l'utilitaire *ping6* envoie plusieurs paquets successifs que le destinataire doit renvoyer (dans l'exercice précédent, l'option `-c1` limitait explicitement le nombre de paquets à 1). On constate que le délai est variable et que l'utilitaire calcule le délai minimum (ici 38,4 ms), le délai moyen (43,6 ms) et le délai maximum (52,7 ms).

Remarque

Le paramétrage de l'utilitaire *ping* définit le nombre de paquets à envoyer, la taille des données qu'il transporte, l'intervalle entre deux paquets… Il est donc facile de créer une attaque malveillante (il existe même des logiciels téléchargeables sur Internet) qui enverrait un grand nombre de paquets de très grande taille, séparés par des intervalles de temps très brefs. Nous verrons au chapitre 10 sur la sécurité que, de ce fait, bien des machines filtrent les requêtes ICMP pour éviter de passer leur temps à répondre à des *ping*.

Exercice 12 : commande traceroute

Solution

1. La première ligne correspond au réseau local dans lequel se trouve l'utilisateur, le premier paquet avec une durée de vie 1 a été détruit par le routeur de sortie du réseau. Il est donc normal que le délai soit très faible.
2. Les astérisques correspondent à des paquets qui se sont perdus, à l'aller ou au retour : au-delà d'un certain délai, on les considère comme manquants.
3. Les délais varient car rien n'est garanti dans l'interconnexion : il peut se produire des « embouteillages » momentanés et/ou des pannes qui provoquent des changements de route.
4. Pour connaître le nombre de réseaux traversés, il suffit de calculer l'adresse réseau de chaque routeur et de compter le nombre de réseaux différents. On en dénombre 10, comme l'illustre le tableau 5.2.

Tableau 5.2 : Les réseaux traversés

193.51.91.1	193.51.91.0 (réseau 1)
2.0.0.1	2.0.0.0 (réseau 2)
11.6.1.1	11.0.0.0 (réseau 3)
11.6.13.1	11.0.0.0 (réseau 3)
189.52.80.1	189.52.0.0 (réseau 4)
193.48.58.41	193.48.58.0 (réseau 5)
193.48.53.49	193.48.53.0 (réseau 6)
193.220.180.9	193.220.180.0 (réseau 7)
195.48.58.43	195.48.58.0 (réseau 8)
195.48.58.50	195.48.58.0 (réseau 8)
194.206.207.18	194.206.207.0 (réseau 9)
194.207.206.5	194.207.206.5 (réseau 10)

5. On ne peut pas connaître les protocoles utilisés au-delà d'IP.

Exercice 13 : traduction d'adresses NAT et PAT

Solution

1. Le NAT statique offre l'accès à Internet à une machine, même si elle possède une adresse privée. Faisant toujours la même correspondance, on donne ainsi la possibilité à une machine d'être vue sur Internet, ce qui est intéressant pour héberger des services vus de l'extérieur. En revanche, cette solution n'apporte aucune amélioration du point de vue de la pénurie des adresses.

2. Le NAT dynamique offre l'accès à Internet à plusieurs machines, mais on ne peut pas les joindre depuis Internet puisque leur adresse n'est pas fixe. L'intérêt évident est la sécurité.
3. Dans ce cas, le routeur prévoit de modifier aussi le numéro de port de la seconde machine et de lui substituer un autre numéro qu'il choisit lui-même. Ce mécanisme s'appelle *PAT* (*Port Address Translation*).

Remarque

Il ne faut pas confondre le mécanisme NAT/PAT avec le *Port Forwarding* ou le *Port Triggering*. Le premier est destiné à rediriger les paquets provenant de l'extérieur qui portent un numéro de port particulier vers une machine spécifique. Une règle spécifie ce traitement dans le routeur. Le second est l'ouverture conditionnelle d'un port, uniquement si l'application en a besoin. Cette technique est intéressante pour les applications manipulant plusieurs numéros de ports, le second port n'étant ouvert que lorsqu'une action précise a eu lieu sur le premier, par exemple.

Exercice 14 : décodage de paquet IP

Solution

`45` 4 = protocole IP version 4 ; 5 = longueur de l'en-tête du paquet = 5 × 4 = 20 octets = longueur par défaut d'un en-tête sans option.

`00` *Type of Service* = 0 = pas de service particulier (en fait, avec IPv4, il n'y a pas de service particulier. Ce champ est donc toujours nul, et s'il n'était pas nul, il serait ignoré !).

`00 50` Longueur totale = 0 × 4096 + 0 × 256 + 5 × 16 + 0 × 1 = 80 octets, donc la longueur du contenu du champ de données est de 80 – 20 = 60 octets.

`20 61` Identifiant du paquet (ne sera utile que s'il est fragmenté).

`00 00` Drapeaux et déplacement = tout à zéro = paquet non fragmenté, *a priori* normal puisqu'il n'y a que 60 octets de données.

`80` Durée de vie = 80 = 8 × 16 + 0 × 1 = 128 routeurs que le paquet pourrait encore traverser.

`01` Protocole transporté dans le paquet : 1 = code du protocole ICMP.

`C5 64` Bloc de contrôle d'erreur de l'en-tête.

`C7 F5 B4 0A` Adresse IP émetteur = `199.245.180.10`.

`C7 F5 B4 09` Adresse IP destinataire = `199.245.180.9`.

Les deux machines sont dans le même réseau de classe *C*, le réseau `199.245.180.0`.

-------Fin de l'en-tête IP----------

Pour décoder le contenu du paquet, il faut connaître le format d'un message ICMP.

`08` Type : 8

`00` Code : 0

L'ensemble type = 8 et code = 0 signifie demande d'écho (correspond à la requête appelée *ping*).

`00 1C` Bloc de contrôle d'erreur sur l'en-tête du message ICMP.

-----Fin de l'en-tête ICMP----------

Contenu quelconque destiné à être renvoyé par le destinataire s'il répond à cette demande d'écho :

```
01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20
21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30
31 32 33 34 35 36 37 38
```

Longueur du contenu ICMP = 56 octets.

---------Fin du contenu ICMP------

--------- Fin du contenu IP----------

Bilan

Le paquet est au format IPv4. Il a été émis par la machine d'adresse IP `199.245.180.10` vers la machine d'adresse IP `199.245.180.9`. Ces deux machines sont dans le même réseau de classe *C*, le réseau `199.245.180.0`. Le paquet possède une longueur totale de 60 octets. Il transporte une requête ICMP de demande d'écho dont la longueur du contenu est de 56 octets : l'émetteur envoie un *ping* au récepteur pour connaître son état.

Exercice 15 : encapsulation d'un paquet IP dans une trame Ethernet

Solution

----------DébutdunetrameEthernet----------

`AA AA AA AA AA AA AA AB` Préambule de synchronisation et délimiteur de début.

`08 00 02 4B 01 C3` *@MAC* destinataire (constructeur = 08 00 02 = 3Com).

`08 00 02 4B 02 D6` *@MAC* émetteur (même constructeur).

`08 00` Type (ici *IP*). (Si la valeur est inférieure à 1 500, c'est une longueur.)

[Ici `08 00` = 2048 ; cette valeur ne peut donc pas être la longueur des données de la trame.]

----------46<=contenu(icipaquetIP)<=1500----------

Le contenu de cette trame est le *ping* de l'exercice précédent.

----------Finducontenu----------

`5F A6 8C 04` Bloc de contrôle d'erreur Ethernet.

Bilan

Cette trame Ethernet a été capturée dans le réseau de classe *C* 199.245.180.0. Deux machines sont concernées par cet échange : la machine *X* d'adresse MAC 08 00 02 4B 02 D6 et d'adresse IP 199.245.180.10 qui a envoyé une requête d'écho (*ping*) à la machine *Y* d'adresse MAC 08 00 02 4B 01 C3 et d'adresse IP 199.245.180.9, située sur le même réseau local. Les cartes Ethernet sont du même constructeur. Les protocoles utilisés sont IP et ICMP.

Exercice 16 : autre exemple

Solution

----------Débutd'unetrameEthernet----------

`FF FF FF FF FF FF` Adresse MAC destinataire (diffusion).

`00 04 80 5F 68 00` Adresse MAC émetteur.

`08 06` Type (ici ARP).

----------46≤Contenu(icimessageARP)≤1500----------

Pour interpréter le contenu de cette trame, il faut disposer du format d'un message ARP.

`00 01` Type de matériel : 1 = Ethernet.

`08 00` Type de protocole : IP.

`06` Longueur de l'adresse physique : 6 octets (pour Ethernet).

`04` Longueur de l'adresse logique : 4 octets (pour IP).

`00 01` Code opération : 1 = Requête ARP.

`00 04 80 5F 68 00` Adresse MAC source.

`89 C2 A2 03` Adresse IP source : 137.194.162.3.

`00 00 00 00 00 00` Adresse MAC destination (vide, car c'est l'adresse qu'on cherche).

`89 C2 A2 F3` Adresse IP destination : 137.194.162.243.

----------Finducontenuréel----------

`00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`

Bourrage (le contenu de la trame est trop court).

----------Finducontenu(46octets)----------

Bilan

Dans un réseau de classe *B* 137.194.0.0, deux machines sont concernées par cette trame : les machines d'adresse IP 137.194.162.3 et 137.194.162.243. La machine 137.194.162.3 envoie une requête ARP en diffusion pour demander l'adresse physique de la machine d'adresse IP 137.194.162.243. La requête ARP est un message court (trop court pour une trame Ethernet) : elle est complétée avec 18 octets de bourrage.

Remarque

Le travail de décodage peut être fait par un équipement particulier appelé *analyseur de protocole*. Configuré avec les types de protocoles utilisés, l'analyseur fournit les informations brutes, découpées champ par champ. Le bilan, quant à lui, est le résultat de la réflexion humaine !

Exercice 17 : adressage IPv6

Solution

1. Cette adresse ne présente aucune suite de valeurs qui peut être compactée.
2. Ici, trois couples d'octets sont entièrement nuls ; on pourra les remplacer par un seul chiffre 0. Ensuite, les 2 octets `0008` sont tels que les trois premiers `0` (situés à gauche) sont inutiles ; de même dans `0800`, le premier `0` est inutile. L'écriture compacte est donc `[1080:0:0:0:8:800:200C:417A]` et, en utilisant le `::`, on obtient `[1080::8:800:200C:417A]`.
3. Cette adresse contient 96 bits à zéro : il s'agit du préfixe d'encapsulation d'une adresse IPv4 dans une adresse IPv6. Elle s'écrit sous forme compacte `[0:0:0:0:0:0:9143:8D5C]`. En utilisant le `::`, on obtient `[::9143:8D]`. L'adresse IPv4 est ici en hexadécimal `91 43 8D 5C`, soit en notation décimale pointée `145.67.141.92`.

Remarque

La notation décimale pointée est quelquefois encore utilisée. On écrira `[::145.67.141.92]`, ce qui affiche directement et lisiblement l'adresse IPv4.

4. `[20A1:0DB8:0000:0000:0000:00A0:1428:57BA]` se compacte en supprimant les zéros à gauche des blocs de 2 octets et en notant `0` globalement pour 2 octets nuls : `[20A1:DB8:0:0:0:A0:1428:57BA]`. En utilisant le `::`, on obtient `[20A1:DB8::A0:1428:57BA]`.
5. L'adresse `3B01:5A8:52AB::/48` représente le réseau dont les 48 premiers bits sont ceux du réseau et les 60 bits suivants ceux des machines. Les adresses des machines vont depuis `3B01:5A8:52AB:0:0:0:0:0` jusqu'à `3B01:5A8:52AB:FFFF:FFFF:FFFF:FFFF:FFFF`.

Chapitre 6

Le routage

Nous avons vu que l'acheminement des messages à travers un ou plusieurs réseaux nécessitait des connaissances sur le réseau et sur l'état de ses liaisons. Les routeurs organisent cet acheminement en exécutant les protocoles de routage. Ils utilisent à cet effet des algorithmes classiques de recherche du meilleur chemin dans un graphe. Dans un grand réseau, chaque routeur construit sa table de routage grâce aux informations de contrôle échangées avec les autres. Pour chaque destination, il consulte sa table, calcule le chemin et son coût, selon des critères prédéfinis dans le protocole.

Nous présentons les deux grandes familles d'algorithmes de routage utilisés dans Internet, avant d'évoquer les particularités du routage pour les réseaux sans fil et les réseaux *peer-to-peer*.

Problèmes et exercices

Exercice 1 : table de routage

Solution

1. Voici un exemple de la table de routage du nœud E : routage(E) = [(A, B) ; (B, B) ; (C, B) ; (D, D) ; (E, –) ; (F, F)], où le couple (A, B) signifie : « Pour aller à A, il faut passer par B. »

   Il existe deux chemins de longueur 2 pour aller de E à A : celui qui passe par B et celui qui passe par F. Nous avons retenu celui qui correspond à la plus petite lettre dans l'ordre alphabétique. Il en est de même pour le chemin de E à C.

2. Avec un algorithme à état des liens, il faut comparer les différents chemins. Le chemin E-B-A est de coût 7 + 2 = 9, alors que E-F-A est de coût 3 + 4 = 7. Ce dernier est meilleur. Remarquons qu'un chemin long comme E-F-D est meilleur que le chemin direct E-D puisque 3 + 3 = 6 est meilleur que 7. L'algorithme de Dijkstra doit donc explorer tous les chemins.

   On procède par étapes. Cherchons les chemins de longueur 1. On trouve E-B = 2, E-D = 7, E-F = 3. Cherchons maintenant les chemins plus longs à partir du lien le plus prometteur, c'est-à-dire E-B. On trouve E-B-A = 2 + 7 = 9 et E-B-C = 2 + 5 = 7. Cherchons ensuite les chemins plus longs à partir du lien prometteur suivant, c'est-à-dire E-F. On trouve E-F-A = 3 + 4 = 7, meilleur que l'information précédemment calculée : cette dernière est effacée, on ne conserve que le meilleur chemin. De même, E-F-D = 3 + 3 = 6 est meilleur que E-D précédemment calculé à 7. On continue ainsi en explorant les chemins à partir du lien prometteur suivant, ici E-C, etc.

   La table de routage de E est finalement : routage(E) = [(A, F) ; (B, B) ; (C, B) ; (D, F) ; (E, –) ; (F, F)].

Exercice 2 : routage avec RIP

Solution

1. À l'état initial, chaque routeur ne connaît que ses voisins directs. La table de routage de A est donc :

   routage(A) = [(A, 0, –) ; (B, 1, B) ; (E, 1, E)].

   De même, les tables de routage des autres routeurs sont :

   routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (E, 1, E)].

   routage(C) = [(B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].

routage(D) = [(C, 1, C) ; (D, 0, –) ; (F, 1, F)].

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

À la date T = 30 s, chaque routeur envoie sa table à ses voisins. Nous traiterons les événements dans l'ordre alphabétique.

Quand A envoie routage(A) = [(A, 0, –) ; (B, 1, B) ; (E, 1, E)] à B, celui-ci constate l'absence de toute nouvelle entrée ; pour celle déjà existantes, les informations envoyées par A ne sont pas plus intéressantes que les siennes. Il en est de même pour E quand il reçoit routage(A) = [(A, 0, –) ; (B, 1, B) ; (E, 1, E)].

Maintenant traitons l'envoi par B de routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (E, 1, E)] à ses voisins A, C et E.

A apprend que B connaît une route de distance 1 pour atteindre C ; il ajoute donc dans sa table une nouvelle entrée (C, 2, B) : C est à une distance 1 de B ou B est à une distance 1 de A, donc C est globalement à une distance 1 + 1 = 2 en passant par B. Les autres informations envoyées par B ne changent rien, la nouvelle table de A est maintenant :

routage(A) = [(A, 0, –) ; (B, 1, B) ; (**C, 2, B**) ; (E, 1, E)].

C apprend que B connaît une route de distance 1 pour atteindre A ; il ajoute donc dans sa table une nouvelle entrée (A, 2, B) : A est à une distance 1 de B ou B est à une distance 1 de C, donc A est globalement à une distance 1 + 1 = 2 en passant par B. Les autres informations envoyées par B ne changent rien ; la nouvelle table de C est maintenant :

routage(C) = [(**A, 2, B**) ; (B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)].

Pour E, l'envoi de B n'apporte rien, sa table reste inchangée.

Maintenant traitons l'envoi par C de routage(C) = [(B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)] à ses voisins B, D, E et F. (Remarque : il s'agit bien de la table initiale de C et non de celle qui a été mise à jour ci-dessus ; par contre, les mises à jour se font sur les tables déjà modifiées.)

B apprend que C connaît une route de distance 1 pour atteindre D et F ; il ajoute donc dans sa table deux nouvelles entrées (D, 2, C) et (F, 2, C). Les autres informations envoyées par C ne changent rien ; la nouvelle table de B est maintenant :

routage(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (**D, 2, C**) ; (E, 1, E) ; (**F, 2, C**)].

La nouvelle table de D devient :

routage(D) = [(**B, 2, C**) ; (C, 1, C) ; (D, 0, –) ; (**E, 2, C**) ; (F, 1, F)].

Il en est de même pour E et F :

routage(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (**D, 2, C**) ; (E, 0, –) ; (F, 1, F)].

routage(F) = [(**B, 2, C**) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)].

L'envoi par D de sa table routage(D) = [(C, 1, C) ; (D, 0, –) ; (F, 1, F)] n'apporte rien à C ni à F.

L'envoi par E de sa table routage$(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (E, 0, –) ; (F, 1, F)]$ provoque l'apparition de l'entrée F dans la table de A et de l'entrée A dans la table de F :

routage$(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (E, 1, E) ; (F, 2, E)]$.

routage$(F) = [(F, 2, E) ; (B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)]$.

Enfin, l'envoi par F de sa table de routage$(F) = [(C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)]$ ne change rien.

À la fin de cette étape, les tables de routage sont donc les suivantes :

routage$(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (E, 1, E)]$.

routage$(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (D, 2, C) ; (E, 1, E) ; (F, 2, C)]$.

routage$(C) = [(A, 2, B) ; (B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)]$.

routage$(D) = [(B, 2, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ; (F, 1, F)]$.

routage$(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)]$.

routage$(F) = [(F, 2, E) ; (B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)]$.

On constate que les chemins de longueur 2 sont maintenant connus.

À la date $T = 1$ min, chaque routeur envoie de nouveau sa table à ses voisins. Sans reprendre le détail des opérations, il est aisé de voir que les tables seront mises à jour avec les chemins de longueur 3. Si on traite les opérations comme précédemment dans l'ordre alphabétique, les tables sont :

routage$(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (D, 3, B) ; (E, 1, E) ; (F, 3, B)]$.

routage$(B) = [(A, 1, A) ; (B, 0, –) ; (C, 1, C) ; (D, 2, C) ; (E, 1, E) ; (F, 2, C)]$.

routage$(C) = [(A, 2, B) ; (B, 1, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)]$.

routage$(D) = [(A, 3, C) ; (B, 2, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ; (F, 1, F)]$.

routage$(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)]$.

routage$(F) = [(A, 2, E) ; (B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)]$.

À la date $T = 1{,}5$ min, chaque routeur envoie de nouveau sa table à ses voisins sans la moindre mise à jour : l'algorithme a convergé.

Remarque

Les routeurs se bornent à connaître leurs voisins immédiats. Ainsi, pour *A*, les destinations *B*, *C*, *D* et *F* sont accessibles en passant par *B* et la destination *E* est directement accessible.

2. La liaison CE tombe en panne ; les tables sont alors les suivantes :

routage$(A) = [(A, 0, –) ; (B, 1, B) ; (C, 2, B) ; (D, 3, B) ; (E, 1, E) ; (F, 3, B)]$.

routage$(B) = [(A, 1, A) ; (B, 0, –) ; (C, 16, C) ; (D, 16, C) ; (E, 1, E) ; (F, 16, C)]$.

routage$(C) = [(A, 16, B) ; (B, 16, B) ; (C, 0, –) ; (D, 1, D) ; (E, 1, E) ; (F, 1, F)]$.

routage$(D) = [(A, 3, C) ; (B, 2, C) ; (C, 1, C) ; (D, 0, –) ; (E, 2, C) ; (F, 1, F)]$.

routage$(E) = [(A, 1, A) ; (B, 1, B) ; (C, 1, C) ; (D, 2, C) ; (E, 0, –) ; (F, 1, F)]$.

routage$(F) = [(A, 2, E) ; (B, 2, C) ; (C, 1, C) ; (D, 1, D) ; (E, 1, E) ; (F, 0, –)]$.

B et *C* signalent à leurs voisins les destinations injoignables. Par conséquent, ceux-ci modifient leurs tables de routage :

routage(*A*) = [(*A*, 0, –) ; (*B*, 1, *B*) ; (**C**, 16, **B**) ; (**D**, 16, **B**) ; (*E*, 1, *E*) ; (**F**, 16, **B**)].

routage(*D*) = [(**A**, 16, **C**) ; (**B**, 16, **C**) ; (*C*, 1, *C*) ; (*D*, 0, –) ; (*E*, 2, *C*) ; (*F*, 1, *F*)].

routage(*E*) = [(*A*, 1, *A*) ; (*B*, 1, *B*) ; (*C*, 1, *C*) ; (*D*, 2, *C*) ; (*E*, 0, –) ; (*F*, 1, *F*)].

routage(*F*) = [(*A*, 2, *E*) ; (**B**, 16, **C**) ; (*C*, 1, *C*) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 0, –)].

On constate que le routeur *E* n'est pas concerné.

À la prochaine échéance de mise à jour régulière (intervalle de 30 s), la diffusion des nouvelles tables va donner lieu à de multiples mises à jour. En particulier, quand *E* transmet sa table, les autres routeurs apprennent l'existence des chemins *EB* et *EC* qui vont se substituer au chemin *BC* défaillant.

routage(*A*) = [(*A*, 0, –) ; (*B*, 1, *B*) ; (**C**, 2, **E**) ; (**D**, 3, **E**) ; (*E*, 1, *E*) ; (**F**, 2, **E**)].

routage(*B*) = [(*A*, 1, *A*) ; (*B*, 0, –) ; (**C**, 2, **E**) ; (**D**, 3, **C**) ; (*E*, 1, *E*) ; (**F**, 2, **E**)].

routage(*C*) = [(**A**, 2, **E**) ; (**B**, 2, **E**) ; (*C*, 0, –) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 1, *F*)].

routage(*D*) = [(*A*, 16, *C*) ; (*B*, 16, *C*) ; (*C*, 1, *C*) ; (*D*, 0, –) ; (*E*, 2, *C*) ; (*F*, 1, *F*)].

routage(*E*) = [(*A*, 1, *A*) ; (*B*, 1, *B*) ; (*C*, 1, *C*) ; (*D*, 2, *C*) ; (*E*, 0, –) ; (*F*, 1, *F*)].

routage(*F*) = [(*A*, 2, *E*) ; (**B**, 2, **E**) ; (*C*, 1, *C*) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 0, –)].

À l'échéance suivante, l'information parvient à *D* et les tables sont alors :

routage(*A*) = [(*A*, 0, –) ; (*B*, 1, *B*) ; (*C*, 2, *E*) ; (*D*, 3, *E*) ; (*E*, 1, *E*) ; (*F*, 2, *E*)].

routage(*B*) = [(*A*, 1, *A*) ; (*B*, 0, –) ; (*C*, 2, *E*) ; (*D*, 3, *C*) ; (*E*, 1, *E*) ; (*F*, 2, *E*)].

routage(*C*) = [(*A*, 2, *E*) ; (*B*, 2, *E*) ; (*C*, 0, –) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 1, *F*)].

routage(*D*) = [(**A**, 3, **C**) ; (**B**, 3, **C**) ; (*C*, 1, *C*) ; (*D*, 0, –) ; (*E*, 2, *C*) ; (*F*, 1, *F*)].

routage(*E*) = [(*A*, 1, *A*) ; (*B*, 1, *B*) ; (*C*, 1, *C*) ; (*D*, 2, *C*) ; (*E*, 0, –) ; (*F*, 1, *F*)].

routage(*F*) = [(*A*, 2, *E*) ; (*B*, 2, *E*) ; (*C*, 1, *C*) ; (*D*, 1, *D*) ; (*E*, 1, *E*) ; (*F*, 0, –)].

À l'échéance suivante, il n'y a plus de changement, l'algorithme a de nouveau convergé.

Exercice 3 : routage avec OSPF

Solution

1. L'état initial est le suivant :

   états(*A*) = [(*B*, 3, *B*) ; (*E*, 5, *E*)].

   états(*B*) = [(*A*, 3, *A*) ; (*C*, 4, *C*) ; (*E*, 1, *E*)].

   états(*C*) = [(*B*, 4, *B*) ; (*D*, 5, *D*) ; (*E*, 2, *E*) ; (*F*, 2, *F*)].

   états(*D*) = [(*C*, 5, *C*) ; (*F*, 1, *F*)].

   états(*E*) = [(*A*, 5, *A*) ; (*B*, 1, *B*) ; (*C*, 2, *C*) ; (*F*, 5, *F*)].

   états(*F*) = [(*C*, 2, *C*) ; (*D*, 1, *D*) ; (*E*, 5, *E*)].

   Les routeurs diffusent (de manière contrôlée) leurs informations à l'intérieur du SA. Chacun peut reconstituer la cartographie du système (voir tableau 6.1).

Tableau 6.1 : Matrice des états des liens du système autonome

État	A	B	C	D	E	F
A	0	3	*x*	*x*	5	*x*
B	3	0	4	*x*	1	*x*
C	*x*	4	0	5	2	2
D	*x*	*x*	5	0	*x*	1
E	5	1	2	*x*	0	5
F	*x*	*x*	2	1	5	0

Dans cette matrice symétrique, les zéros sur la diagonale indiquent un coût nul ; les *x* repèrent les liaisons inexistantes. Calculons la table de routage du routeur *A*.

A connaît deux routes *AB* de coût 3 en passant par *B* directement et *AE* de coût 5 en passant par *E* directement. Il place la route *AE* en attente (5 est moins bon que 3) et valide la route *AB* à partir de laquelle il explore les chemins passant par *B*.

A	*AB*, 3, *B*	*x*	*x*	*AE*, 5, *E*	*x*

La route *BC* est de coût 4, donc *AC* sera de coût 3 + 4 = 7 en passant par *B*. La destination *C* n'était pas encore connue ; la route est ajoutée dans la table.

La route *BE* est de coût 1, donc *AE* sera de coût 3 + 1 = 4 en passant par *B* ce qui est meilleur que la route actuellement connue. La table est mise à jour.

A	*AB*, 3, *B*	*AC*, 7, *B*	*x*	*AE*, 4, *B*	*x*

Le routeur *A* place la route *AC* en attente (7 est moins bon que 4) et valide maintenant la route *AE* à partir de laquelle il explore les chemins passant par *E*(*).

Les routes *EA* et *EB* n'apportent rien : ni nouvelle destination, ni route meilleure.

La route *EC* est de coût 2, donc *AC* sera de coût 4 + 2 = 6 en passant par *B* ce qui est meilleur que la route actuellement connue. La table est mise à jour.

La route *EF* est de coût 5, donc *AF* sera de coût 4 + 5 = 9 en passant par *E*. La destination *F* n'était pas encore connue ; la route est ajoutée dans la table.

A	*AB*, 3, *B*	*AC*, 6, *E*	*x*	*AE*, 4, *B*	*AF*, 9, *E*

Le routeur *A* place la route *AF* en attente (9 est moins bon que 6) et valide maintenant la route *AC* à partir de laquelle il explore les chemins passant par *C*.

Les routes *CB* et *CE* n'apportent rien : ni nouvelle destination, ni route meilleure.

La route *CD* est de coût 5, donc *AD* sera de coût 6 + 5 = 11 en passant par *C*. La destination *D* n'était pas encore connue ; la route est ajoutée dans la table.

La route *CF* est de coût 2, donc *AF* sera de coût 6 + 2 = 8 en passant par *C* ce qui est meilleur que la route actuellement connue. La table est mise à jour.

A	*AB*, 3, *B*	*AC*, 6, *E*	*AD*, 11, *C*	*AE*, 4, *B*	*AF*, 8, *C*

Le routeur *A* place la route *AD* en attente (11 est moins bon que 8) et valide maintenant la route *AF* à partir de laquelle il explore les chemins passant par *F*.

Les routes *FC* et *FE* n'apportent rien.

La route *FD* est de coût 1, donc *AD* sera de coût 8 + 1 = 9 en passant par *F* ce qui est meilleur que la route actuellement connue. La table est mise à jour.

A	*AB*, 3, *B*	*AC*, 6, *E*	*AD*, 9, *F*	*AE*, 4, *B*	*AF*, 8, *C*

La route *AD* est validée à son tour, et les routes *DC* et *DE* n'apportent rien.

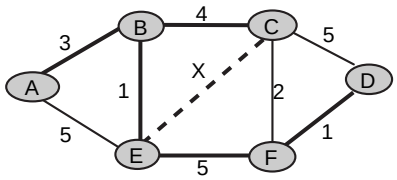
L'algorithme est terminé.

La table de routage de *A* est alors la suivante :

routage(*A*) = [(*A*, 0, –) ; (*B*, 3, *B*) ; (*C*, 6, *B*) ; (*D*, 9, *B*) ; (*E*, 4, *B*) ; (*F*, 8, *B*)].

Toutes les routes passent par le routeur *B*, comme l'illustre la figure 6.1.

Figure 6.1
Système autonome avec les meilleures routes en gras.



2. Le lien entre *C* et *E* tombe en panne. Les routeurs *C* et *E* diffusent l'information dans le système. La matrice du système est maintenant illustrée au tableau 6.2.

Tableau 6.2 : Matrice des états des liens du système autonome après la panne E-C

État	A	B	C	D	E	F
A	0	3	*x*	*x*	5	*x*
B	3	0	4	*x*	1	*x*
C	*x*	4	0	5	*x*	2
D	*x*	*x*	5	0	*x*	1
E	5	1	*x*	*x*	0	5
F	*x*	*x*	2	1	5	0

L'application de l'algorithme par le routeur *A* est la même que précédemment jusqu'à l'étape marquée (*) dans la correction de la question 1 et que nous reprenons ci-après.

A	*AB*, 3, *B*	*AC*, 7, *B*	*x*	*AE*, 4, *B*	*x*

Le routeur *A* place la route *AC* en attente (7 est moins bon que 4) et valide maintenant la route *AE* à partir de laquelle il explore les chemins passant par *E*(*).

Les routes *EA* et *EB* n'apportent rien. La route *EF* est de coût 5, donc *AF* sera de coût 4 + 5 = 9 en passant par *E*. La destination *F* est ajoutée.

A	*AB*, 3, *B*	*AC*, 7, *B*	*x*	*AE*, 4, *B*	*AF*, 9, *E*

Le routeur *A* place la route *AF* en attente et valide la route *AC* à partir de laquelle il explore les chemins passant par *C*. On obtient :

A	*AB*, 3, *B*	*AC*, 7, *B*	*AD*, 12, *C*	*AE*, 4, *B*	*AF*, 9, *E*

et enfin :

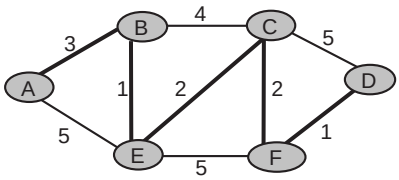
A	*AB*, 3, *B*	*AC*, 7, *B*	*AD*, 10, *F*	*AE*, 4, *B*	*AF*, 9, *E*

La nouvelle table de routage de A est maintenant :

routage(A) = [(A, 0, –) ; (B, 3, B) ; (C, 7, B) ; (D, 10, E) ; (E, 4, B) ; (F, 9, E)].

La figure 6.2 illustre les routes retenues.

Figure 6.2
Routes recalculées après la panne.



Exercice 4 : routage multicast dans un réseau

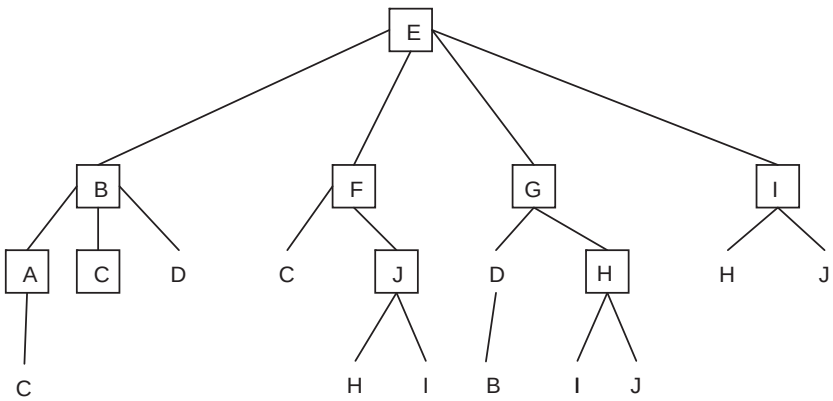
Solution

1. L'arbre collecteur de E est représenté à la figure 6.3. *Explications* : E envoie un paquet aux routeurs B, F, G et I (premier saut, représenté par la première ligne de l'arbre). Par hypothèse, ces nœuds sont sur le chemin privilégié pour atteindre E. Au deuxième saut, cinq paquets arrivent par des chemins privilégiés. Au troisième, les paquets émis par les différents routeurs sont considérés comme dupliqués et rejetés par leurs destinataires.

Remarque

Ce mécanisme fournit un critère d'arrêt de la diffusion : elle est terminée lorsque tous les nœuds-feuilles de l'arbre (les nœuds du dernier saut) ne renvoient aucun paquet, puisqu'ils viennent par des chemins non privilégiés.

Figure 6.3
Exemple d'arbre collecteur construit par *E*. Les nœuds appartenant au chemin privilégié pour l'atteindre sont encadrés par un carré.



2. Le tableau 6.3 donne le nombre de paquets produits.

Tableau 6.3 : Nombre de paquets produits à chaque saut

N° du saut	Nombre de paquets
1er saut	4
2e saut	9
3e saut	6

Il y a donc 4 + 9 + 6 = 19 paquets émis en tout.

3. Par inondation, chaque routeur envoie un paquet vers tous les autres, donc un paquet sur chaque lien. En supposant que chaque routeur n'envoie qu'un seul paquet vers les routeurs voisins, un minimum de 27 paquets seront envoyés sur les 14 liaisons existantes. Cette valeur est minimale car, compte tenu des hypothèses actuelles, on ne dispose d'aucun critère d'arrêt pour stopper la diffusion ! Il faut envisager le cas où les routeurs envoient systématiquement un paquet reçu sur tous les autres liens (sauf sur celui d'où il provient). Les paquets tournent alors sans fin dans le réseau.

Remarque

La prolifération anarchique des messages est un phénomène redouté dans un réseau, car il faut tout arrêter. Cet accident peut se produire avec un arbre recouvrant, lorsque des boucles sont créées dans la topologie (par suite d'erreurs de branchement ou parce que les messages de mises à jour sont synchronisés avec les modifications de la topologie qui en découle). Il en résulte ce que l'on appelle des *data storms* (tempêtes de données) dans les réseaux de commutateurs que nous verrons au chapitre 7.

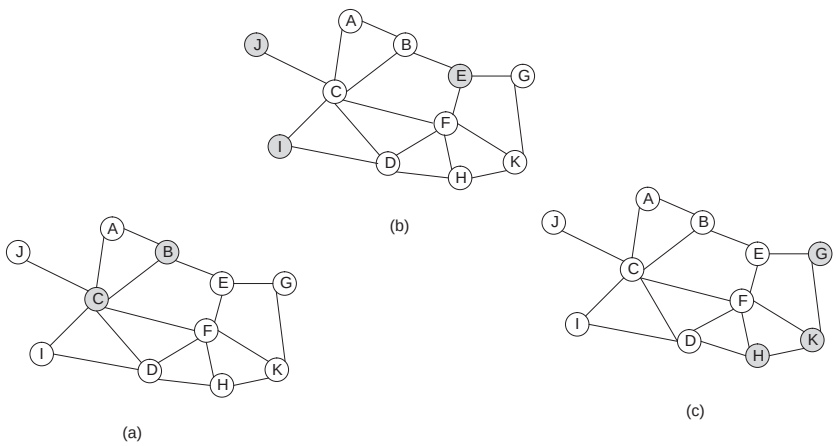
Exercice 5 : routage dans un réseau de mobiles *ad hoc*

Solution

1. La figure 6.4 illustre la diffusion du paquet *ROUTE REQUEST* émis par *A*.

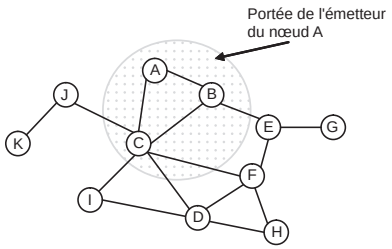
Figure 6.4

Diffusion du paquet de recherche de la route pour aller à *K*. Les nœuds encore inconnus sont indiqués en grisé sur les figures. (a) Découverte de nouveaux nœuds après le premier saut. (b) Découverte des nouveaux nœuds après le deuxième saut. (c) Découverte des nouveaux nœuds après le troisième saut.



2. Comme l'illustre la figure 6.4(c), le nœud *K* est découvert par *F* au troisième saut. La diffusion ne s'arrête qu'au quatrième saut, lorsque tous les nœuds ayant reçu le paquet de recherche de route ne le diffusent plus.
3. La nouvelle topologie est donnée à la figure 6.5.

Figure 6.5
Nouvelle topologie du réseau.

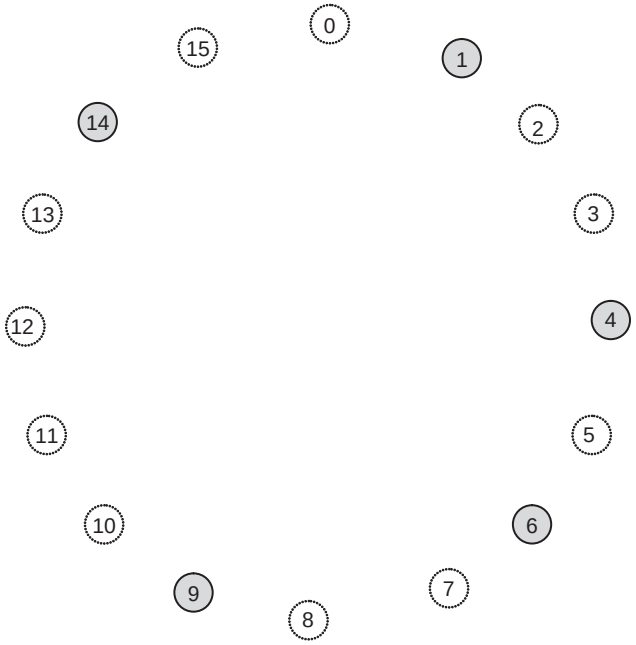


Exercice 6 : routage dans un réseau P2P (*peer-to-peer*)

Solution

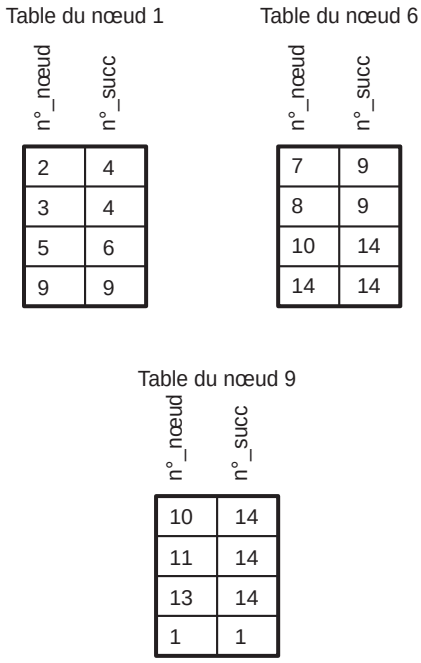
1. Le cercle comprend 16 valeurs, numérotées de 0 à 15, comme indiqué à la figure 6.6.

Figure 6.6
Représentation circulaire des identifiants avec $m = 4$. Les nœuds non utilisés sont représentés en pointillé, les nœuds réels sont en gris.



2. Les tables de repères des trois nœuds sont représentées par la figure 6.7.

Figure 6.7
Contenu des tables de repères des nœuds 1, 6 et 9.



3. Les résultats de la fonction *successeur(k)* pour les nœuds 4, 10,14, 22 et 28 sont respectivement : *successeur(4)* = 10 ; *successeur(10)* = 14 ; *successeur(22)* = 22 ; *successeur(28)* = 1.
4. Le nœud d'identifiant 39 correspond à 7, car la représentation se fait modulo 2^m.

Remarque

La représentation avec de petites valeurs de *m* que nous utilisons à des fins pédagogiques entraîne de nombreuses homonymies si l'on veut répertorier un grand nombre de valeurs, d'où des difficultés pour identifier le nœud auquel on s'adresse effectivement. On comprend pourquoi l'algorithme de Chord utilise 160 bits, de façon à minimiser le risque d'homonymes. Avec 160 bits, on peut identifier 2^{160} valeurs différentes, soit environ 10^{48} (en arrondissant 2^{10} à 10^3).

5. Les valeurs des entrées des trois tables de repères sont données aux tableaux 6.4, 6.5 et 6.6.

Tableau 6.4 : Table de repères du nœud 1

n°_nœud	*n°_succ*
2	4
3	4
5	7
9	12
17	20

Tableau 6.5 : Table de repères du nœud 4

n°_nœud	*n°_succ*
5	7
6	7
8	12
12	12
20	20

Tableau 6.6 : Table de repères du nœud 12

n°_nœud	*n°_succ*
13	15
14	15
16	20
20	20
28	1

6. Les tableaux 6.7 et 6.8 donnent les valeurs connues des tables de repères des nœuds 4 et 12.

Tableau 6.7 : Table de repères partiellement remplie du nœud 4

n°_nœud	*n°_succ*
5	7
6	7
8	–
12	–
20	–

Tableau 6.8 : Table de repères partiellement remplie du nœud 12

n°_nœud	*n°_succ*
13	15
14	15
16	–
20	–
28	–

7. *Premier cas : recherche du successeur du nœud 3.* Il s'agit du cas le plus simple : 1 sait que 3 précède son propre successeur (4). Le nœud recherché est 4 et la recherche s'arrête là. Le nœud 1 fait ensuite la correspondance entre l'identifiant de nœud et l'adresse IP recherchée.

   Deuxième cas : recherche du successeur du nœud 14. Par définition, 14 ne se trouve pas dans l'intervalle [1, 4]. En consultant sa table de repères, le nœud 1 constate que le prédécesseur le plus proche est 9 (quatrième entrée). Le successeur réel de 9 est 12.

Le nœud 1 envoie une requête à l'adresse IP du nœud 12. Celui-ci voit que 14 se situe entre lui et son successeur (il appartient à l'intervalle [12, 15]). 12 envoie en réponse à la requête du nœud 1 l'adresse IP correspondant au nœud 15.

Troisième cas : recherche du successeur du nœud 19. Le nœud 1 voit que 19 n'appartient pas à l'intervalle entre lui et son successeur immédiat. Il envoie donc une requête au nœud 4. Celui-ci constate de même que 19 se trouve au-delà de son successeur (7). Il propage donc la requête au nœud 7, qui procède de même puisque 19 ne se trouve pas non plus dans l'intervalle compris entre lui et son successeur [12, 15] : il envoie la requête reçue à 15. Celui-ci constate que 19 se situe bien dans l'intervalle [15, 20]. Il envoie comme réponse au nœud 12 l'adresse IP de 20. Celle-ci passe successivement aux nœuds 7 et 4 avant de revenir au nœud 1.

Au passage de la réponse fournie au nœud 1, les nœuds contactés complètent les entrées suivantes de leur table de repères :

– Pour le nœud 1 : les deux premières entrées sont déjà remplies. Il complète sa table avec les couples suivants :

  < 5, 7 > (3e entrée) et < 9, 12 > (4e entrée).

– Pour le nœud 4 : aucune entrée n'est remplie. Il complète sa table avec les couples suivants :

  < 5, 7 > (1re entrée) ; < 6, 7 > (2e entrée) ; < 8, 12 > (3e entrée) ; < 12, 12 > (4e entrée).

– Pour le nœud 7 : les trois premières entrées sont déjà remplies, mais il ne peut pas obtenir davantage d'informations sur ses successeurs.

– Pour le nœud 12 : les deux premières entrées sont déjà remplies, mais il ne peut pas obtenir davantage d'informations sur ses successeurs.

Chapitre 7

Interconnexion de réseaux et réseaux d'entreprise

Puisque les réseaux locaux ont des portées limitées, un réseau d'entreprise se constitue de plusieurs réseaux locaux reliés soit par des équipements spécifiques, soit par des liaisons ou des réseaux grande distance. Nous examinerons comment relier les différents types de réseaux grâce aux équipements disponibles, puis nous verrons l'évolution des infrastructures, notamment les implications de l'utilisation croissante d'Internet dans l'architecture des réseaux. Nous terminerons par un rapide survol des pratiques des usagers.

Problèmes et exercices

Exercice 1 : répéteur, pont et routeur

Solution

Tableau 7.1 : Comparaison entre répéteur, pont et routeur

Éléments de comparaison	Répéteur	Pont	Routeur
Simplicité d'installation et de configuration	Oui	Oui	Non
Niveau d'interconnexion	1	2	3
Filtrage	Non	Sur les adresses MAC	Sur les adresses IP
Trafic de service	Non	Non, sauf *Spanning Tree Algorithm*	Important *Routing Information Protocol*
Suppression des collisions	Non	Oui	Oui
Protocoles traités	Aucun	Indépendant des protocoles	Une version de logiciel par protocole traité
Temps de traitement interne	Nul	Faible	Important
Évolutivité	Aucune	Faible	Grande
Filtrage du trafic de diffusion	Non	Non	Oui
Gestion de la sécurité du réseau	Non	Non	Oui

Exercice 2 : rôle des ponts et des commutateurs

Solution

1. Si *E* et *H* sont des ponts, ils reçoivent l'un et l'autre la trame émise par *F*. Les deux ponts, ne sachant pas où se trouve le destinataire, laissent passer la trame, laquelle circule sur les deux autres segments. *C* reçoit donc la trame. *K*, situé dans le troisième segment, la voit passer et peut l'enregistrer.
2. Lorsque *C* répond à *F*, la trame diffusée parvient au pont *E* qui sait qu'il doit la laisser passer puisque *F* est de l'autre côté. Par contre, le pont *H* sait que *C* et *F* sont accessibles sur le même port. Il filtre donc la trame qui ne transite pas sur le troisième segment : l'analyseur de protocole *K* ne voit pas la trame réponse.
3. Si *D*, *E*, *I* et *H* sont des répéteurs, le réseau ne peut pas fonctionner car il n'a plus sa structure de bus ramifié. Si *D*, *E*, *I* et *H* sont des ponts, le réseau ne peut fonctionner que si l'un des ponts est inactif (ce qui a pour effet de « couper » la boucle). Les ponts constituent un ensemble collaboratif, ils discutent entre eux et décident celui qui sera inactif (c'est le rôle de STP, l'algorithme de l'arbre recouvrant).

Remarque

L'explication de la question 1 suppose que les ponts fassent de l'auto-apprentissage. Si les ponts sont configurés à l'avance avec les adresses MAC de toutes les stations, seul le pont *D* laisse passer la trame qui va de *F* à *C* ainsi que sa réponse de *C* à *F*. L'analyseur *K* ne voit rien de ce trafic. Cet exemple montre l'intérêt des ponts – ou des commutateurs – pour segmenter des réseaux locaux.

Exercice 3 : architecture réseau d'une petite entreprise

Solution

1. Il n'y a qu'un seul domaine de diffusion : les commutateurs font circuler les trames dans tout le réseau de l'entreprise, depuis les postes de travail jusqu'aux serveurs et *vice versa*.
2. Chaque commutateur du cœur de réseau ne voit que trois commutateurs (et non six), puisqu'une pile de commutateurs est considérée et gérée comme une seule entité logique.
3. Les ruptures de liens provoquent une boucle dans le réseau si le protocole STP n'est pas activé. La boucle se produit entre : Étage 1 – *C1* – *C2* – Étage 2 – Étage 3 – *C1* – É tage 1.
4. Le réseau dans l'état actuel n'est évidemment pas connecté, puisqu'il ne contient aucun équipement de routage vers Internet. Le plus simple est de remplacer les commutateurs par des commutateurs-routeurs et d'obtenir un accès à Internet auprès d'un fournisseur d'accès (FAI).
5. Puisque chaque VLAN constitue un sous-réseau, il faut utiliser la fonction de routage des commutateurs-routeurs pour passer d'un VLAN à l'autre. Il y a donc cinq domaines de diffusion.
6. Actuellement, une station de travail appartient à un seul VLAN, vraisemblablement celui de l'étage dans lequel elle se trouve. Ce regroupement géographique est également fonctionnel : l'appartenance à un étage dépend de la fonction occupée au sein de l'entreprise. Si les modes de travail évoluaient, l'administrateur du réseau pourrait facilement installer des VLAN supplémentaires pour constituer des groupes de travail interétages ou encore répartir les trois VLAN sur les différents étages.
7. On veut éviter que le trafic de transit entre étages perturbe l'accès aux serveurs et aux imprimantes. Par ailleurs, cette structure limite l'envoi de requêtes ARP des stations de travail vers les imprimantes ou les serveurs, puisqu'une station n'a besoin que de l'adresse MAC de son routeur (au lieu des adresses MAC du serveur ou de l'imprimante choisis). Notons toutefois que cette « amélioration » se paie par un accroissement de la charge de routage des commutateurs-routeurs.

8. Le réseau fonctionne sans problème, mais il ne dispose d'aucune protection contre des attaques malveillantes. Il faut des sécurités (au moins un pare-feu !), que nous verrons au chapitre 10.

Remarques

1. L'architecture réseau présentée ici est une structure très répandue, car facile à installer et à maintenir. En outre, elle offre une évolutivité qui tient compte de l'activité et du développement de l'entreprise.
2. Aucune hypothèse n'est faite sur la répartition de la charge entre les deux commutateurs-routeurs. On peut supposer que les deux sont actifs en même temps et gèrent des trafics indépendants. Si l'on souhaite un fonctionnement actif/passif pour conserver un réseau en état de marche même en cas de défaillance d'un des commutateurs-routeurs, il faut ajouter le protocole VRRP.

Exercice 4 : intérêt des protocoles antiboucles

Solution

1. L'en-tête d'un paquet IP contient un champ *Durée de vie* (ou *TTL*), décrémenté à chaque passage dans un routeur. Dès que la valeur de ce champ s'annule, le paquet correspondant est éliminé. Les routeurs ne provoqueront jamais de prolifération anarchique des messages.

   Pour commuter les trames d'un port à l'autre, les commutateurs – comme les ponts – utilisent l'en-tête d'une trame Ethernet. Malheureusement, celui-ci ne contient aucun champ *Durée de vie*. Un pont est donc incapable de savoir si la trame qu'il reçoit sur un port est la première du genre ou la 2 500e ! Il retransmet indéfiniment la même trame sur tous ses ports, sauf sur le port d'où elle provient. De plus, comme l'on ne dispose d'aucun critère pour l'arrêter automatiquement, cette inondation accidentelle peut durer éternellement si personne n'intervient…
2. Seules deux possibilités sont envisageables pour supprimer les boucles : soit une intervention manuelle sur site pour déconnecter les liens physiques, soit une déconnexion logique des ports redondants si l'on peut encore administrer les ponts à distance.
3. Dans un cas comme dans l'autre, l'intervention d'un technicien (évident pour une déconnexion manuelle !) est nécessaire. En effet, même avec une déconnexion à distance, un technicien peut être obligé d'intervenir si la perte d'un lien a rendu un équipement inaccessible. L'intérêt des protocoles STP et RSTP réside dans l'automatisation de la suppression des boucles.
4. Les protocoles STP et RSTP font appel à plusieurs temporisateurs pour faire basculer les ports des commutateurs d'un état dans un autre. Des temporisateurs mal calibrés ou une BPDU retardée malencontreusement peuvent engendrer une instabilité du réseau et déclencher les *data storms* redoutées (notons que le bouclage est un phénomène beaucoup moins probable avec RSTP, en raison de la rapidité de convergence de cet algorithme).

Exercice 5 : algorithme de l'arbre recouvrant

Solution

1. Avant le transfert des données, les commutateurs exécutent l'algorithme STP pour éviter les boucles dans le réseau. Une fois l'arbre recouvrant calculé, il n'y a plus qu'un seul chemin pour que les données circulent entre *Eth1* et *Eth2*.
2. Certains commutateurs ne reçoivent plus de BPDU de configuration, ce qui provoque une nouvelle exécution de l'algorithme pour construire une autre arborescence.
3. Il ne reçoit plus de BPDU de configuration sur le port en panne. Il démarre de manière intempestive une nouvelle exécution de STP. Comme il ne peut plus recevoir les BPDU des autres commutateurs, il peut déclencher une tempête de diffusion (*broadcast storm*). Il faut le déconnecter !
4. La meilleure BPDU de configuration qu'il puisse produire est : `< 15.4.27 >`.
5. La valeur de l'ID doit être au maximum de 14.
6. Le port racine est le port 2. Les ports 1 et 2 font partie de l'arbre recouvrant.

Exercice 6 : intérêt de la pondération de liens

Solution

1. Considérons les chemins possibles entre la racine et tous les commutateurs. Puisque STP cherche le moindre coût pour atteindre tous les commutateurs, il trouve des chemins de coût maximal 2 : pour aller à *C2* depuis *C/R1*, le chemin passe par *C1* et, pour aller à *C3* depuis *C/R1*, il suffit de passer par *C/R2*. Les paquets ne passeront jamais par le lien entre *C2* et *C3*.
2. En affectant un coût de 100 au lien entre *C/R2* et *C3*, on force l'utilisation d'un seul lien en fonctionnement normal. Ce choix peut se justifier si le trafic entre *C1*, *C2* et *C3* est prépondérant (il y a peu d'accès vers *C/R1* ou *C/R2*).

Exercice 7 : arbre recouvrant avec pondération des liens

Solution

1. Les coûts entre les différents segments sont récapitulés au tableau 7.2.

Tableau 7.2 : Coûts affectés à tous les liens

Pont	**1**		**2**		**3**		**4**		**5**	
Port	1	2	1	2	1	2	1	2	1	2
Coût	1	10	1	10	1	10	10	10	1	10

2. Les liens entre : le commutateur 3 et *Eth1*, le commutateur 5 et *Eth3* seront bloqués.
3. Les BPDU émises sur leurs différents ports sont récapitulées au tableau 7.3, dans lequel BPDU_*i* représente la meilleure BPDU de configuration fabriquée par le nœud *i*.

Tableau 7.3 : Meilleures BPDU émises sur les différents ports des commutateurs

N° commut.	**N° port**	**BPDU émise**	**Commentaires**
1	1	`1.0.1`	La racine envoie sa BPDU
	2	`1.0.1`	La racine envoie sa BPDU
2	1	`1.0.1`	Port racine : retransmet BPDU_1
	2	`1.1.2`	Pont désigné sur *Eth3* : transmet BPDU_2
3	1	`1.0.1`	Port racine : retransmet BPDU_1
	2	`1.1.3`	Pont désigné sur *Eth4* : transmet BPDU_3
4	1	`1.1.3`	Port bloqué : propage BPDU_3
	2	`1.10.4`	Port racine : retransmet BPDU_1
5	1	`1.10.5`	Port racine : retransmet BPDU_1
	2	`1.10.5`	Port bloqué : propage BPDU_2

Voici quelques explications supplémentaires :

- *Commutateur 4* : le port 2 est bloqué car le coût vers la racine est nettement plus élevé que par l'autre chemin, bien que ce commutateur soit directement relié à la racine
- *Commutateur 5* : le port 2 est bloqué car STP a privilégié le chemin par le commutateur 2.
- Le standard 8023.1D préconise des valeurs à titre indicatif. Les valeurs des coûts de l'exercice sont arbitraires mais simples à calculer.

Exercice 8 : VLAN par ports

Solution

1. Lorsque *M1* envoie une trame avec l'adresse de diffusion, le commutateur la répète sur l'ensemble de ses ports : tous les équipements, de *M1* à *M6*, la reçoivent. Quand *M1* envoie ensuite une trame à *M3*, le commutateur la reçoit sur le port *P1* et la transmet sur le port *P3* : seul *M3* la reçoit.
2. Le commutateur associe *P1*, *P3* et *P5* au VLAN *A* et *P2*, *P4* et *P6* au VLAN *B*.
3. Le commutateur diffuse au sein du VLAN *A* la trame de *M1* arrivant par le port *P1*. Les équipements de numéros pairs ne la reçoivent pas : le commutateur isole les équipements des deux VLAN, le trafic de l'un ne passe pas sur l'autre. Le traitement de la trame envoyée par *M1* à *M3* est inchangé, puisque *M1* et *M3* sont à l'intérieur du même VLAN *A*.
4. Le second commutateur peut disposer d'une table semblable à celle du premier : *P1*, *P3* et *P5* appartiennent au VLAN *A* et *P2*, *P4* et *P6* au VLAN *B*. Il reste à relier les deux commutateurs. À cet effet, on peut relier les ports 7 de chaque commutateur et affecter ce port au VLAN *A*. De même, on relie les ports 8 de chacun d'eux et on les affecte au VLAN *B*.

   La table devient alors : *P1*, *P3*, *P5* et *P7* au VLAN *A* et *P2*, *P4*, *P6* et *P8* au VLAN *B*.
5. Si on met deux liens (en reliant les deux ports *P7* entre eux et les deux ports *P8* entre eux), on obtient une boucle à éliminer par STP. Un seul lien entre les deux commutateurs (entre les ports *P7*, par exemple) ne crée pas de boucle, mais les deux VLAN doivent alors partager ce lien : le port *P7* doit appartenir aux deux VLAN. Dans ce cas, il faut que les VLAN soient étiquetés pour que les commutateurs sachent traiter les trames. Il faut pour cela que les cartes réseau des équipements supportent le standard d'étiquetage des VLAN 802.1Q.

Remarque

Cet exercice montre l'intérêt du protocole 802.1Q : en permettant la création de plusieurs arbres recouvrants, on relie les deux commutateurs par plusieurs trunks, ce qui améliore la tolérance aux pannes.

Exercice 9 : VRRP pour équilibrer le trafic dans un réseau

Solution

1. Pour introduire une redondance dans l'accès aux VLAN, il faut doubler le dispositif qui assure l'acheminement dans le réseau : le plus simple est d'installer deux commutateurs-routeurs utilisant le protocole VRRP pour raccorder chaque serveur par deux

liens différents (en *double attachement* : un lien est actif pendant que l'autre est désactivé). Les deux commutateurs-routeurs constituent le cœur du réseau de la figure 7.16 du livre.

2. Avec l'adjonction du nouveau commutateur, on a créé la boucle : Serveur 1 – commutateur 1 – serveur 2 – commutateur 2 – serveur 1. C'est la raison pour laquelle les liens sont saturés.
3. Pour supprimer la boucle, il est indispensable, avant toute chose, d'activer STP. Pour privilégier un chemin en fonctionnement normal sans interdire la redondance, il faut utiliser des arbres recouvrants multiples, en construisant un STP par VLAN selon le standard 802.1Q.

Remarque

La solution précédente est à compléter car, en cas de défaillance d'un équipement, le chemin entre un VLAN donné et le serveur auquel il fait le plus souvent appel n'est pas forcément simple. On a donc intérêt à ajouter un lien entre commutateur 1 et commutateur 2 pour simplifier le trajet dans le réseau.

Chapitre 8

Les protocoles de transport

Avant de décrire les protocoles de transport qui diffèrent par bien des aspects, nous présentons les concepts qui leur sont communs : les notions de ports et de *sockets*.
Le service fourni par IP n'étant pas fiable, le choix d'un protocole de transport dépend de la qualité de service dont les applications ont besoin. TCP (*Transport Control Protocol*) est employé pour les échanges exigeant une grande fiabilité. UDP (*User Datagram Protocol*) est préféré par les applications moins exigeantes, qui se contentent d'un service de bout en bout en mode sans connexion. Enfin, nous présentons succinctement les principes des protocoles allégés fonctionnant pour des applications à contraintes de temps strictes.

Problèmes et exercices

Exercice 1 : principes et intérêt de TCP

Solution

1. Nous avons noté que le débit maximal sur Ethernet était de 9,82 Mbit/s si le débit réel était de 10 Mbit/s (voir exercice 7, chapitre 4). On supposait, pour faire le calcul, que les 1 500 octets de la trame étaient des octets utiles. Si le champ de données de la trame Ethernet transporte un paquet IP contenant un segment TCP, il y a (sauf options) 20 + 20 = 40 octets d'en-tête, soit seulement 1 460 octets de données utiles. Le débit maximal est donc de $10 \times (1\,460/1\,528) = 9{,}55$ Mbit/s.
2. Puisque tous les segments TCP ont le même format d'en-tête (demande d'ouverture de connexion, segment de données ou fermeture de connexion), le traitement est toujours le même et peut être optimisé pour une meilleure efficacité.
3. L'intérêt est de disposer d'un contexte, mémorisé chez l'émetteur comme chez le destinataire (protocole de bout en bout), dans lequel sont conservés tous les paramètres fixes et variables de la connexion : il est ainsi possible de suivre l'évolution de la connexion, d'adapter au mieux les délais pour la mise en œuvre des fonctions de contrôle d'erreur, de contrôle de flux, de séquencement et de congestion.
4. L'ordre des paquets IP n'étant pas géré par IP (les paquets n'ont pas de numéro de séquence !), TCP reçoit les données des paquets IP et les réordonne (seulement dans le cas où tous les fragments sont arrivés). TCP assure ce service en numérotant les octets du flot de données d'un segment.
5. Oui, bien sûr ! Et même plusieurs centaines simultanément… Par exemple, une application simple comme la navigation sur le Net ouvre (sans que l'utilisateur le sache…) des dizaines de connexions : chaque objet multimédia dans la page consultée correspond à une connexion ; chaque clic engendre l'ouverture d'une nouvelle connexion… Une connexion est identifiée par deux sockets (numéro de port local, adresse IP locale ; numéro de port distant, adresse IP distante). Même si l'adresse IP locale est la même, le numéro de port change : il correspond au processus que crée le système d'exploitation de la machine local. Les numéros distants peuvent varier de même. Il ne peut donc jamais y avoir confusion entre deux connexions, même si les adresses IP sont les mêmes, puisque les processus ont des identifiants différents.

Exercice 2 : identification de plusieurs connexions TCP

Solution

1. Le logiciel TCP n'existe que dans les postes des utilisateurs (clients ou serveurs). La couche IP constitue la couche de niveau supérieur des routeurs.
2. TCP a la capacité de gérer plusieurs connexions simultanément. Plusieurs connexions sont donc envisageables entre *PC1* et *PC2*. Ces connexions diffèrent par le numéro de port local et par le numéro de port distant ; il n'y a donc pas de confusion possible. Les deux sockets valent respectivement : < adresse *IP-PC1*, port *x,* adresse *IP-PC2*, port 80 > et < adresse *IP-PC1*, port *y*, adresse *IP-PC2*, port 21 >.
3. La nouvelle connexion avec le service Web utilise des numéros de séquence pour les octets du flot de données échangées qui sont différents de la connexion précédente, puisque le numéro de séquence initial est tiré au sort pour la nouvelle connexion. Il n'existe par conséquent aucun risque que des segments interfèrent.

Exercice 3 : traitement d'un segment TCP

Solution

1. L'application émettrice (sur la machine *A*, port *x*) demande à TCP l'ouverture d'une connexion avec l'application de *B*, port *y*. TCP fabrique donc un segment d'ouverture de connexion avec (port *x*, port *y*, SYN = 1), placé dans un paquet IP avec *A* comme adresse IP émetteur et *B* comme adresse IP destinataire.
2. Que les deux machines soient dans le même réseau ou non ne change rien au fonctionnement de TCP. Seul le traitement fait à l'interface entre IP et les couches inférieures change.

   Dans le premier cas, le paquet est encapsulé dans une trame du réseau local avec *A* comme adresse MAC émetteur et *B* comme adresse MAC destinataire. Dans le second cas, le paquet est encapsulé une première fois dans une trame du réseau local de *A* avec *A* comme adresse MAC émetteur et *RA* (routeur qui gère la sortie du réseau) comme adresse MAC destinataire. Il est encapsulé une seconde fois dans une trame du réseau local de *B* avec *RB* (routeur qui gère l'entrée dans le réseau) comme adresse MAC émetteur et *B* comme adresse MAC destinataire.

   Lorsque la machine *B* reçoit le paquet, elle en extrait le segment et avertit le module TCP de l'arrivée d'informations correspondant à un échange provenant d'une machine d'adresse `IP A`. Le module TCP analyse le segment et prévient l'application identifiée par le port *y* de la demande d'ouverture de connexion.

Exercice 4 : statistiques de connexions TCP

Solution

La machine dispose de sept connexions TCP actives. Sur l'ensemble des connexions gérées, le trafic est à peu près symétrique (autant de segments reçus que de segments émis…), sous réserve que la taille des données soit équivalente. La qualité de la transmission est bonne puisqu'on observe 102 retransmissions pour 107 736 segments envoyés (107 838 = 107 736 + 102) ; le taux d'erreur sur les segments est de $9{,}5 \times 10^{-4}$. Si l'on est dans un réseau Ethernet avec des segments de 1 460 octets utiles, le taux d'erreur est d'environ 2×10^{-8}.

Exercice 5 : statistiques détaillées de connexions TCP

Solution

1. 352 383 segments (appelés ici *paquets* !) ont transporté 489 293 303 octets. La taille moyenne d'un segment est 489 293 303/352 383 = 1 388 octets (ou 1,355 Ko).
2. La cause la plus vraisemblable des pertes est la non-fiabilité des réseaux traversés : TCP attend les acquittements un certain temps et décide que les segments sont perdus quand l'acquittement n'est pas arrivé.
3. Les paquets `window probe` et `window update` sont destinés à modifier la taille de la fenêtre pour un meilleur contrôle de flux et de congestion.
4. Les paquets de contrôle sont ceux qui correspondent à l'ouverture et à la fermeture des connexions ainsi qu'aux acquittements de ces ouvertures et fermetures.
5. Sur les 224 123 segments reçus, 204 197 contiennent des acquittements. La proportion est de :

   204 197/224 123 = 91 %

6. Chaque acquittement reçu valide la réception par l'autre extrémité d'un certain nombre d'octets. Les 204 197 acquittements ont validé globalement 487 292 645 octets, soit 487 292 645/204 197 = 2 386 octets (ou 2,33 Ko).
7. Les 17 100 segments reçus en séquence correspondent aux données que la machine a reçues correctement. Le trafic est dissymétrique puisque la machine reçoit peu de données et énormément d'acquittements.
8. La taille moyenne des segments de données reçus est de 2 306 201/17 100 = 135 octets. La taille des segments émis est de 1,355 Ko, soit 10 fois plus. La machine sur laquelle ces statistiques sont effectuées est un serveur, qui reçoit de la part de ses nombreux clients des requêtes simples et courtes et émet en réponse des messages longs. Ce pourrait être un serveur Web ou un serveur FTP, par exemple.

9. La machine a reçu 356 demandes d'ouverture de connexion et en a accepté seulement 92. Par conséquent, (356 – 92)/356 = 75 % de tentatives de connexion ont échoué. Cette observation pourrait accréditer l'hypothèse d'un serveur FTP pour lequel un *login* et un mot de passe sont nécessaires pour ouvrir une connexion.
10. Le volume moyen de données émis par connexion est de 489 293 303/92 = 5 318 405 octets, soit 5 Mo. Le volume moyen de données reçu par connexion est de 2 306 201/92 = 24 Ko. Cela confirme le trafic dissymétrique de la machine.
11. 176 379 mises à jour du paramètre RTT ont eu lieu sur les 204 197 acquittements reçus. On voit que 86,3 % des acquittements correspondent à des acquittements normaux.
12. Le temporisateur a expiré 21 fois parce que l'acquittement n'arrivait pas. On en conclut que le RTT est bien adapté aux différentes connexions.
13. Aucune fermeture de connexion n'est survenue, grâce au mécanisme *Keepalive* qui surveille le bon fonctionnement de la connexion en l'absence de trafic. L'activité sur les connexions établies est satisfaisante.

Exercice 6 : décodage de segment TCP

Solution

`00 15` -> Port source, ici 21, donc serveur FTP.

`0F 87` -> Port destination 3975, port quelconque du client.

`9C CB 7E 01` -> Numéro de séquence (numéro du 1er octet émis).

`27 E3 EA 01` -> Numéro de séquence (numéro du 1er octet attendu en réception).

`5` -> Longueur de l'en-tête du segment (20 octets).

`0 12 = 0000 0001 0010` -> Drapeaux.

```
      |_____|||  ||||_ FIN (clôture de la connexion) = 0.
         |   ||   |||_ SYN (ouverture – ou réponse d'ouverture – de connexion) = 1.
         |   ||  ||_   RST (réinitialisation de la connexion) = 0.
         |   ||  |_    PSH (livraison immédiate) = 0.
         |   ||____    ACK (accusé de réception) = 1 ; le segment transporte un
                       accusé de réception.
         |   |_____    URG (urgent) = 0.
         |_________    6 bits réservés.
```

Les drapeaux SYN et ACK sont mis à 1.

`10 00` -> Taille de la fenêtre, ici *a priori* 4 096 octets. C'est la quantité de données que l'émetteur est autorisé à envoyer sans accusé de réception.

`DF 3D` -> Bloc de contrôle d'erreur sur le segment entier.

`00 00` -> Pointeur vers les données urgentes (nul ici, puisqu'il n'y a pas de données urgentes ; bit URG = 0).

Fin du segment TCP (sans données).

Bilan

Ce segment contient deux drapeaux mis à 1 : les drapeaux SYN et ACK. Il s'agit donc de la réponse positive à une demande d'ouverture de connexion. Les numéros de séquence initiaux viennent d'être choisis, et le serveur, qui accepte la demande d'ouverture de connexion du client, fixe la taille de la fenêtre à 4 Ko. Il s'agit d'un serveur FTP, identifié par l'utilisation de son port bien connu, 21. À l'analyse du segment seul, on ne sait quelles sont les machines concernées par l'échange. On sait juste qu'il s'agit de l'initialisation d'un transfert de fichier FTP entre un client et un serveur.

Exercice 7 : décodage complet d'une trame

Solution

----------Début d'une trame Ethernet ----------

`AA AA AA AA AA AA AA AB` -> Synchronisation (préambule et début de trame).

`08 00 20 0A 70 66` -> Adresse MAC destinataire (constructeur = `080020`).

`08 00 20 0A AC 96` -> Adresse MAC émetteur (carte de même constructeur).

`08 00` -> Type (ici IP).

---------Début du contenu de la trame de longueur = 1500 octets (ici datagramme IP)---------

`4` -> Version du protocole IP (IPv4).

`5` -> Longueur de l'en-tête (5 × 32 bits = 160 bits ou 5 × 4 octets = 20 octets).

`00 00 28 A6 F5 00 001A 06 75 94 C0 5D 02 01 84 E3 3D 05` -> En-tête IP.

| |__| |__| |___| |_| |_| |___| |________| |_________|

| | | | | | | | |_ @IP destinataire 132.227.61.5.

| | | | | | | |_ @IP émetteur 192.92.2.1.

| | | | | | |_Bloc de contrôle d'erreur (sur l'en-tête du datagramme seulement).

| | | | | |_ Protocole (ici TCP).

| | | | |_ *TTL* (ici 1*A* = 1 × 16 + 10 = 26 routeurs ou secondes).

| | | |_ Drapeau + Déplacement (`0` = inutl ; `0` = *DF* [fragmentation autorisée]) ; 0 = *MF* (pas de fragments à suivre, donc dernier fragment). `0000000000000` = déplacement, c'est-à-dire position du 1er octet du fragment par rapport au 1er octet du paquet initial. Ce fragment est le premier et le dernier du paquet : il s'agit donc d'un paquet non fragmenté.

| | |_ID du paquet (numéro quelconque, ne sert que si le paquet est amené à être fragmenté).

| |_ Longueur totale (ici `00 28` en hexadécimal, soit : 2 × 16 + 8 en décimal donc 40 octets).

|_ Pas de qualité de service (ToS).

--------Contenu du paquet = segment TCP d'une longueur de 20 octets (40-20) ----------

Le segment est celui de l'exercice précédent.

---------- Fin du segment TCP (sans données) ----------

---------- Fin des données du paquet IP ----------

`20 20 20 20 20 20` -> 6 octets de bourrage pour amener la trame Ethernet à la longueur *minimale* (64 octets en tout).

`9B 52 46 43` -> Bloc de contrôle d'erreur de la trame Ethernet.

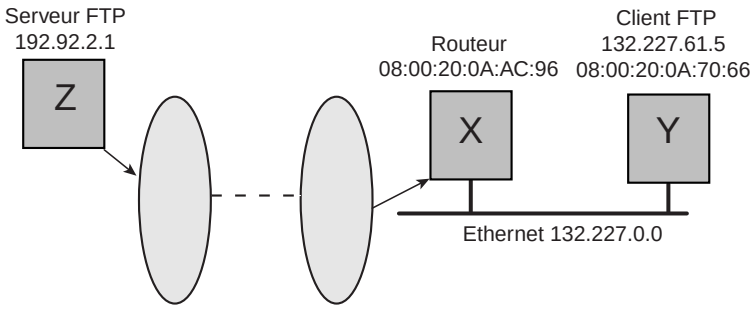
---------- Fin de la trame Ethernet ----------

Bilan

Cette trame Ethernet a été capturée dans un réseau que nous ne connaissons pas forcément. Trois possibilités sont à envisager :

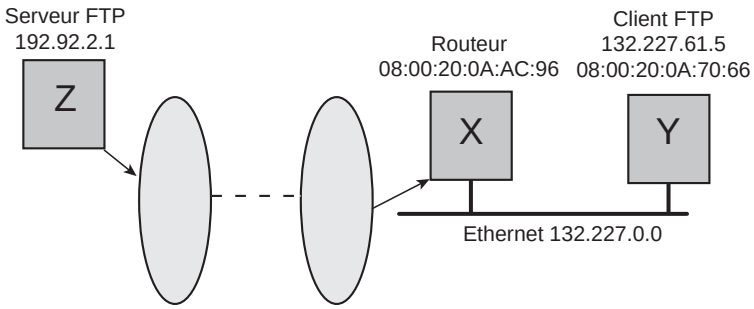
a. Dans le réseau de l'émetteur de la trame (réseau de classe C `192.92.2.0`). Dans cette hypothèse, trois machines sont concernées par cet échange (voir figure 8.1) : la machine *X* d'adresse MAC `08 00 20 0A AC 96` et d'adresse IP `192.92.2.1`. C'est un serveur FTP qui envoie une réponse positive à la demande d'ouverture de connexion que lui a faite la machine *Y*, client FTP d'adresse `IP 132.227.61.5`, se trouvant dans un autre réseau dont nous ne connaissons pas la technologie. La machine *Z* d'adresse MAC `08 00 20 0A 70 66` est le routeur de sortie du réseau `192.92.2.0`. Elle est explicitement désignée comme destinataire de la trame Ethernet, puisque le paquet IP que celle-ci contient doit sortir du réseau ! (Nous n'avons pas à connaître l'adresse IP du routeur.)

Figure 8.1 Machines concernées par l'échange dans l'hypothèse a.



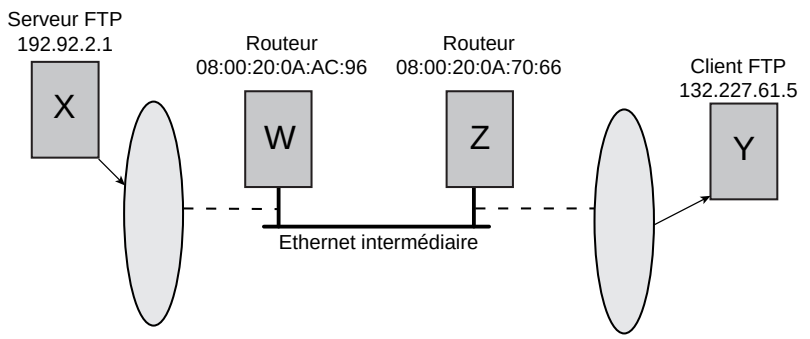
b. *Dans le réseau du destinataire (réseau de classe B* `132.227.0.0`*).* De manière similaire, trois machines sont concernées par cet échange (voir figure 8.2) : la machine X d'adresse MAC `08 00 20 0A AC 96`, qui est le routeur d'entrée du réseau en question. Ce routeur expédie une trame Ethernet à la machine Y, client FTP d'adresse MAC `08 00 20 0A 70 66` et d'adresse IP `132.227.61.5` se trouvant dans le réseau. La trame transporte l'accusé de réception à la demande de connexion faite par la machine Y (contenue dans une trame précédente que nous ignorons). La machine Z d'adresse IP `192.92.2.1` est le serveur FTP situé à l'extérieur du réseau, ce qui justifie que le paquet soit relayé par le routeur d'entrée.

Figure 8.2 Machines concernées par l'échange dans l'hypothèse b.



c. *Dans un réseau de transit, entre les deux réseaux concernés.* La capture a pu être faite dans un réseau intermédiaire qui ne contient ni l'émetteur, ni le destinataire du paquet (voir figure 8.3). Les adresses MAC sont celles de deux routeurs intermédiaires, connus uniquement par leurs adresses MAC. Quatre machines sont donc concernées par cet échange : la machine *X*, la machine *Y* et les routeurs du réseau de transit.

Figure 8.3 Machines concernées par l'échange dans l'hypothèse c.



Exercice 8 : fonctionnement RTP/RTCP

Solution

1. Le numéro ajouté par RTP sert à détecter d'éventuelles pertes dans le réseau (les numéros doivent se suivre). L'estampille temporelle reconstruit l'intégrité temporelle du flux.
2. Compte tenu du choix de l'horloge à 166 Hz, si l'estampille temporelle du premier paquet est *n*, celle du deuxième émis 1/166 de seconde plus tard, donc 1 top d'horloge plus tard, est $n + 1$, celle du troisième $n + 2$, etc.

   Le tableau 8.1 explicite, paquet après paquet, l'estampille temporelle, la date d'arrivée et la date à laquelle le récepteur peut « consommer » en imaginant qu'il démarre aussitôt.

Tableau 8.1 : Dates de réception et de « consommation » des paquets

Paquet n°	Estampille	Date de réception	Date de « consommation »
1	n	$t + 150$	$t + 150$
2	$n + 1$	$t + 156$	$t + 156$
3	$n + 2$	$t + 162$	$t + 162$
4	$n + 3$	$t + 168$	$t + 168$
…			
98	$n + 97$	$t + 976$	$t + 976$
99	$n + 98$	$t + 982$	$t + 982$
100	$n + 99$	$t + 988$	$t + 988$
101	$n + 100$	$t + 1044$	À la date $t + 994$, le paquet n'est pas arrivé : situation de famine !
102	$n + 101$	$t + 1050$	À la date $t + 1\,000$, le paquet n'est pas arrivé : situation de famine !
…			Famine

Si le délai varie dans le réseau, on constate que le décodeur se trouve en situation dite de *famine*. Avec l'estampille temporelle, il reconstruit le flux audio en « consommant » les paquets au rythme indiqué. Malheureusement, un retard dans la transmission empêche cette reconstruction.

La solution pour une bonne reconstitution du flux à la réception nécessite de disposer d'une estimation du délai maximal de traversée du réseau. Ici, on peut supposer que le réseau garantisse un délai maximal de 200 ms. Le récepteur ne démarre la consommation qu'après avoir attendu ce délai maximal : en ajoutant un petit retard initial (insoupçonnable pour les récepteurs humains), le décodeur est sûr d'avoir toujours des données à consommer (ce qu'illustre le tableau 8.2).

Tableau 8.2 : Dates de réception et de « consommation » des paquets pour un délai maximal de 200 ms

Paquet n°	Estampille	Date de réception	Date de « consommation »
1	n	$t + 150$	$t + 200$
2	$n + 1$	$t + 156$	$t + 206$
3	$n + 2$	$t + 162$	$t + 212$
4	$n + 3$	$t + 168$	$t + 218$
…			
98	$n + 97$	$t + 976$	$t + 1\,026$
99	$n + 98$	$t + 982$	$t + 1\,032$
100	$n + 99$	$t + 988$	$t + 1\,038$
101	$n + 100$	$t + 1\,044$	À la date $t + 1\,044$, le paquet vient tout juste d'arriver
102	$n + 101$	$t + 1\,050$	À la date $t + 1\,050$, le paquet vient tout juste d'arriver

Remarque

Pour cet exemple, nous avons choisi, dans un objectif de simplification, un flux à débit constant : un paquet toutes les 6 ms. On constate qu'il est prudent d'attendre encore un peu plus pour le récepteur… Cependant, toute attente supplémentaire remplit les mémoires tampon. Les flux à débit variable sont de ce fait encore plus délicats à traiter. L'estampille temporelle et les indications véhiculées sur le canal RTCP sont très utiles.

Exercice 9 : contrôle de trafic avec un leaky bucket

Solution

La rafale de 2 000 paquets est bloquée par le premier seau : les 1 000 premiers paquets consomment les 1 000 jetons du seau puis patientent en consommant le jeton du second seau. Ils entrent donc dans le réseau au rythme où ce dernier est réalimenté, soit

50 paquets par seconde. Le premier seau se remplissant au même rythme, 50 nouveaux paquets de la source à chaque seconde consomment les 50 jetons générés dans le premier seau. On est donc dans un régime de croisière avec 50 paquets par seconde entrant au niveau du premier seau, 1 000 paquets patientant entre les deux seaux et un taux d'entrée dans le réseau de 50 paquets par seconde.

Une fois que tous les paquets ont passé le premier seau, celui-ci se remplit alors que le second continue à bloquer le trafic à l'entrée du réseau. La rafale de 2 000 paquets entre dans le réseau au rythme du second seau, soit 50 paquets par seconde : au total, 40 s lui sont nécessaires pour entrer.

À la fin, le premier seau contient de nouveau 1 000 jetons et, en l'absence de nouveau trafic à émettre, le premier seau continue à se remplir jusqu'à atteindre sa taille maximale de 1 500 jetons.

Remarques

1. Le leaky bucket est un exemple de méthode de contrôle de trafic destiné à lisser les rafales, typiquement hérité de l'architecture ATM. La taille maximale du premier seau est définie pour limiter la période pendant laquelle l'entrée du réseau est occupée à plein régime.
2. Si le flux traité a des contraintes temporelles fortes, on voit que ce mécanisme, intéressant pour l'opérateur du réseau puisqu'il lisse les rafales de trafic, détruit le rythme existant. Il est indispensable de gérer des estampilles temporelles de RTP pour reconstruire le rythme.

Chapitre 9

Les applications

De très nombreuses applications utilisent TCP/IP. Nous en abordons ici quelques-unes : la configuration dynamique des machines, le service de noms de domaine, le courrier électronique, le transfert de fichiers, sans oublier l'incontournable navigation sur le Web. Les deux premières sont en fait des applications internes, utiles au bon fonctionnement des réseaux ; elles rendent des services indispensables comme la distribution des adresses IP, la correspondance entre noms symboliques et adresses IP des machines. Les autres concernent directement les utilisateurs et leurs besoins de communiquer. Le courrier électronique est l'une des premières applications développées dans Internet. La navigation sur le Web est à l'origine de l'engouement populaire pour Internet.

Problèmes et exercices

Exercice 1 : utilisation de DHCP

Solution

1. Les numéros de port sont 68 pour le client et 67 pour le serveur.
2. La machine n'ayant pas d'adresse IP, elle laisse à « plein 0 » le champ adresse IP source. Comme elle ne connaît pas non plus l'adresse du destinataire, elle utilise l'adresse de diffusion « plein 1 ». On a donc :
   - adresse IP source = `0.0.0.0`, port source 68 ;
   - adresse IP destination = `255.255.255.255`, port destination 67.
3. Le paquet IP est encapsulé dans une trame Ethernet dont l'adresse MAC de destination est `FF:FF:FF:FF:FF:FF` puisqu'il s'agit d'une diffusion. L'adresse MAC source est le numéro de série de la carte réseau de la machine en question.

Exercice 2 : redondance de serveurs DHCP

Solution

1. Une machine du troisième sous-réseau peut obtenir une adresse IP par DHCP, car le routeur implémente un agent relais qui transfère le message en diffusion (`255.255.255.255`) au-delà du routeur. Les deux serveurs DHCP reçoivent alors la requête *DHCPDiscover*.
2. Deux serveurs DHCP ont été installés pour une meilleure fiabilité : il est rare que les deux tombent simultanément en panne…

Exercice 3 : rôle d'un serveur DNS et trafic interne

Solution

La machine *A* doit en premier lieu obtenir du serveur DNS la conversion du nom symbolique *www.soc.pays* en adresse IP. Son fichier de configuration lui fournissant l'adresse IP du serveur DNS à interroger, il doit émettre une requête ARP pour obtenir son adresse MAC.

Trame 1 sur réseau 1 = trame Ethernet diffusée par A (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP (champ protocole = 0805) pour connaître l'adresse MAC du serveur DNS que *A* connaît seulement par son adresse IP. Le serveur DNS qui a reçu cette trame et reconnu son adresse IP répond.

Trame 2 sur réseau 1 = trame Ethernet (@MAC *X* vers @MAC *A*) contenant la réponse ARP fournissant l'adresse MAC du serveur DNS. *A* inscrit dans sa table ARP la correspondance @IP 25.0.1.33 = @MAC 08:00:02:54:E2:A0. Maintenant que *A* connaît l'adresse MAC de *X*, elle peut lui envoyer une trame Ethernet.

Trame 3 sur réseau 1 = trame Ethernet (@MAC *A* vers @MAC *X*). Cette trame contient un paquet IP (@IP *A* vers @IP *X*) qui contient un message UDP (port distant 53), qui contient la requête au DNS : (« Je recherche l'adresse IP de *www.soc.pays*. »)Trame 4 sur réseau 1 = trame Ethernet (@MAC *X* vers @MAC *A*). Cette trame contient un paquet IP (@IP *X* vers @IP *A*). Le paquet contient le message UDP (port local 53) portant la réponse du DNS. (*www.soc.pays* = @IP 25.0.2.55). *A* connaît l'adresse IP de son correspondant, en l'occurrence le serveur Web. En utilisant le masque de sous-réseau présent dans son fichier de configuration, *A* constate que le serveur Web n'est pas dans le même sous-réseau que lui. Il faut passer par le routeur pour sortir de son sous-réseau. Or, le routeur n'est connu (fichier de configuration de *A*) que par son adresse IP, notée ici @ IP *R1* : une nouvelle requête ARP servira à obtenir son adresse MAC.

Remarque

On pourrait se demander pourquoi le fichier de configuration contient l'adresse IP du routeur alors que c'est l'adresse MAC du routeur qui sert. Fournir l'adresse IP est une solution souple qui permet de changer l'équipement matériel du routeur sans avoir à reconfigurer toutes les machines… Nous avons vu au chapitre 7 qu'on utilise aujourd'hui la notion d'adresse IP virtuelle de routeur pour apporter une souplesse supplémentaire.

Trame 5 sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @ MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP pour connaître l'adresse MAC du routeur connu par @IP *R1*.

Trame 6 sur réseau 1 = trame Ethernet (@MAC *R1* vers @MAC *A*). Cette trame contient la réponse ARP fournissant l'adresse MAC du routeur (côté sous-réseau 1). *A* inscrit dans sa table ARP la correspondance @IP 25.0.1.1 = @MAC 08:00:02:54:E2:A2 ; maintenant que *A* connaît l'adresse MAC du routeur (notée @MAC *R1*), il peut lui envoyer une trame Ethernet.

Trame 7 sur réseau 1 = trame Ethernet (@MAC *A* vers @MAC *R1*). Cette trame recèle un paquet IP (@IP *A* vers @IP *W*) qui contient un segment TCP de demande d'ouverture de connexion (drapeau SYN) pour HTTP (port 80).

Le routeur a reçu la trame 7 qui lui était adressée. Il en a extrait le paquet IP et, après consultation de sa table de routage, constate que le réseau de *W* était joignable directement sur sa seconde interface. Nous faisons ici l'hypothèse que l'adresse MAC de *W* ne figure pas dans la table ARP du routeur.

Trame 8 sur réseau 2 = trame Ethernet diffusée par le routeur (@MAC *R2* vers @ MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP pour connaître l'adresse MAC de *W* dont le routeur ne connaît que @IP *W*.

Trame 9 sur réseau 2 = trame Ethernet (@MAC *W* vers @MAC *R2*). Cette trame contient la réponse ARP fournissant l'adresse MAC de *W*. Le routeur inscrit dans sa table ARP la correspondance @IP `25.0.2.55` = @MAC `08:00:02:54:E2:7F`. Maintenant qu'il connaît l'adresse MAC de *W*, il peut lui envoyer une trame Ethernet.

Trame 10 sur réseau 2 = trame Ethernet (@MAC *R2* vers @MAC *W*). Cette trame contient le paquet IP (@IP *A* vers @IP *W*), qui renferme le segment TCP de demande d'ouverture de connexion pour HTTP (port 80). Ce paquet est celui de la trame 7, la seule différence étant le champ TTL que le routeur a réduit de 1. Le bloc de contrôle d'erreur sur l'en-tête a été recalculé.

Trame 11 sur réseau 2 = trame Ethernet diffusée par *W* (@MAC *W* vers @MAC `FF:FF:FF:FF:FF:FF`) contenant une requête ARP pour connaître l'adresse MAC du routeur (*A* est dans un autre sous-réseau).

Trame 12 sur réseau 2 = trame Ethernet (@MAC *R2* vers @MAC *W*). Cette trame contient la réponse ARP fournissant l'adresse MAC du routeur (côté sous-réseau 2).

Trame 13 sur réseau 2 = trame Ethernet (@MAC *W* vers@MAC *R2*). Cette trame contient un paquet IP (@IP *W* vers @IP *A*), qui possède un segment TCP de réponse positive à la demande d'ouverture de connexion (drapeaux SYN et ACK) pour HTTP.

Trame 14 sur réseau 1 = trame Ethernet diffusée par *R1* (@MAC *R1* vers @MAC `FF:FF:FF:FF:FF:FF`). Cette trame contient la requête ARP pour connaître l'adresse MAC de *A*.

Trame 15 sur réseau 1 = trame Ethernet (@MAC *A* vers@MAC *R1*). Cette trame contient la réponse ARP fournissant l'adresse MAC de *A*.

Trame 16 sur réseau 1 = trame Ethernet (@MAC *R1* vers @MAC *A*). Cette trame contient un paquet IP (@IP *W* vers @IP *A*), qui renferme le segment TCP de réponse positive à la demande d'ouverture de connexion pour HTTP. Ce paquet est celui transporté dans le réseau 2, encapsulé dans la trame 13, aux champs TTL et bloc de contrôle d'erreur près.

Trame 17 sur réseau 1 = trame Ethernet (@MAC *A* vers @MAC *R1*). Cette trame contient un paquet IP (@IP *A* vers @IP *W*), lequel recèle un segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP (port 80).

Trame 18 sur réseau 2 = trame Ethernet (@MAC *R2* vers @MAC *W*). Cette trame contient le paquet IP (@IP *A* vers @IP *W*), contenant le segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP (port 80).

Remarque

Cet exercice est délibérément détaillé. L'objectif était de montrer l'ensemble du trafic généré par la recherche des adresses MAC et par l'utilisation du serveur DNS. D'autre part, il illustre la notion d'encapsulation en insistant sur le fait que les requêtes HTTP ou DNS sont transmises dans des messages de la couche 4 (TCP ou UDP), lesquels sont véhiculés par les paquets IP, eux-mêmes constituant le champ de données d'une trame Ethernet. Nous vous proposons, ci-après, une version simplifiée, dans laquelle on suppose que tous les caches ARP contiennent les correspondances nécessaires : on enlève tout le trafic ARP.

Il ne reste que deux phases d'échange : l'interrogation de l'annuaire et l'ouverture de la connexion TCP avec le serveur Web.

a. Interrogation de l'annuaire :

   Trame 3 sur réseau 1 = trame Ethernet (de *A* vers *X*) contenant un paquet IP (de *A* vers *X*) contenant un message UDP contenant la requête au DNS (adresse IP de *W* ?).

   Trame 4 sur réseau 1 = trame Ethernet (de *X* vers *A*) contenant un paquet IP (de *X* vers *A*) contenant un message UDP contenant la réponse du DNS (adresse IP de *W*).

b. Ouverture de connexion TCP en trois temps avec le serveur Web :

   Trame 7 sur réseau 1 = trame Ethernet (de *A* vers *R1*) contenant un paquet IP (de *A* vers *W*) contenant un message TCP de demande d'ouverture de connexion (drapeau SYN) pour HTTP (port 80). On traverse le routeur.

   Trame 10 sur réseau 2 = trame Ethernet (de *R2* vers *W*) contenant le paquet IP (de *A* vers *W*) contenant un message TCP de demande d'ouverture de connexion (drapeau SYN) pour HTTP (port 80).

   Trame 13 sur réseau 2 = trame Ethernet (de *W* vers *R2*) contenant un paquet IP (de *W* vers *A*) contenant un message TCP de réponse positive à la demande d'ouverture de connexion (drapeaux SYN et ACK) pour HTTP. On traverse de nouveau le routeur.

   Trame 16 sur réseau 1 = trame Ethernet (de *R1* vers *A*) contenant un paquet IP (de *W* vers *A*) contenant un message TCP de réponse positive à la demande d'ouverture de connexion (drapeaux SYN et ACK) pour HTTP.

   Trame 17 sur réseau 1 = trame Ethernet (de *A* vers *R1*) contenant un paquet IP (de *A* à *W*) contenant un segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP. On traverse encore le routeur.

   Trame 18 sur réseau 2 = trame Ethernet (de *R2* vers *W*) contenant le paquet IP (de *A* à *W*) contenant le segment TCP de confirmation d'ouverture de connexion (drapeau ACK) pour HTTP (port 80).

Enfin, si on ne donne que la vision applicative, deux échanges ont lieu :

a. Interrogation de l'annuaire : requête de *A* au DNS (adresse IP de *W* ?) et réponse du DNS (adresse IP de *W*).

b. Ouverture de connexion TCP en trois temps avec le serveur Web :
   - demande d'ouverture de connexion de *A* (drapeau SYN) pour *W* (port 80) ;
   - réponse positive à la demande d'ouverture de connexion de *W* (drapeaux SYN et ACK) pour *A* ;
   - confirmation d'ouverture de connexion de *A* (drapeau ACK) pour *W*.

Exercice 4 : serveur DHCP et serveur DNS

Solution

Par rapport à l'exercice précédent, il faut ajouter le trafic lié à la recherche d'un serveur DHCP et le dialogue avec celui-ci.

Trame I sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient un paquet IP (@IP 0.0.0.0 vers @IP 255.255.255.255), qui encapsule un message UDP (port 67) contenant le message *DHCPDiscover*.

Trame II sur réseau 1 = trame Ethernet diffusée par *Y* (@MAC *Y* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient un paquet IP (@IP *Y* vers @IP 172.25.64.75), qui encapsule un message ICMP (*Echo Request*). Le serveur DHCP teste si l'adresse IP qu'il veut proposer à *A* est disponible en envoyant un *ping* à cette adresse. Si l'adresse est disponible, la requête *ping* n'obtient pas de réponse.

Trame III sur réseau 1 = trame Ethernet (@MAC *Y* vers @MAC *A*). Cette trame contient un paquet IP (@IP *Y* vers @IP 255.255.255.255), qui encapsule un message UDP (port 68) contenant le message *DHCPOffer* avec @IP 172.25.64.75.

Trame IV sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient un paquet IP (@IP 0.0.0.0 vers @IP 255.255.255.255), qui encapsule un message UDP (port 67) contenant le message *DHCPRequest* : « J'ai choisi le serveur DHCP *Y* avec son offre @IP 172.25.64.75. »

Trame V sur réseau 1 = trame Ethernet (@MAC *Y* vers @MAC *A*). Cette trame contient un paquet IP (@IP *Y* vers @IP 172.25.64.75), qui encapsule un message UDP (port 68) contenant le message *DHCPAck* : « Voici les autres informations de configuration : masque de sous-réseau, adresse IP routeur par défaut, adresse IP serveur DNS… »

Trame VI sur réseau 1 = trame Ethernet diffusée par *A* (@MAC *A* vers @MAC FF:FF:FF:FF:FF:FF). Cette trame contient une requête ARP : « Je cherche l'adresse MAC de la machine d'adresse IP 172.25.64.75. »

A priori, si tout s'est bien passé (l'adresse IP proposée était libre), la trame VI n'a pas de réponse.

Les trames I à VI sont échangées avant les trames 1 à 18 de l'exercice précédent.

Exercice 5 : enregistrements sur un serveur DNS

Solution

Refresh est l'intervalle de temps entre deux vérifications d'un serveur secondaire sur le serveur officiel primaire pour savoir si des modifications ont été apportées et si une mise à jour est nécessaire ; *Retry* est le temps d'attente d'un serveur secondaire avant de renouveler sa mise à jour si la précédente a échoué ; *Expire* est le temps au bout duquel les informations sont jetées si elles n'ont pas pu être mises à jour ; *TTL* est la durée pendant laquelle un serveur DNS peut conserver en cache un enregistrement du fichier de la base de données.

Les temporisateurs sont exprimés en secondes : 3600 représente une heure ; 21600 = six heures ; 172800 = deux jours et 604800 = une semaine.

Exercice 6 : serveur DNS et cache

Solution

1. Le serveur DNS interroge un serveur racine pour savoir qui gère *pays*. Avec la réponse, il interroge le serveur DNS qui gère *pays* pour savoir qui gère *domaine.pays*. Avec la réponse, il interroge enfin le serveur DNS qui gère *domaine* pour récupérer l'adresse IP de *ftp.domaine.pays*. Il met à jour son cache avec les informations recueillies.
2. Cette fois-ci, les informations du cache sont utiles. Il suffit d'interroger un serveur DNS qui gère *domaine.pays* pour connaître l'adresse du serveur Web *www.domaine.pays*. Le serveur DNS met de nouveau à jour son cache avec les informations recueillies.
3. Le serveur DNS s'adresse au serveur DNS qui gère *pays* (déjà connu) pour savoir qui gère *autre_domaine.pays*, puis il interroge ce dernier pour connaître l'adresse du serveur : *www.autre_domaine.pays.* Il met encore à jour son cache avec les informations recueillies, qui seront utiles pour la requête suivante. Il suffira d'interroger le serveur DNS connu qui gère *autre_domaine.pays* pour connaître l'adresse du serveur DNS qui gère *sous_domaine.autre_domaine.pays* et demander à ce dernier l'adresse du serveur FTP *ftp.sous_domaine. autre_domaine.pays.* Une dernière fois, il met à jour son cache avec les informations recueillies.

Exercice 7 : annuaire LDAP et base de données

Solution

On peut noter trois grandes différences :

- Un annuaire est *a priori* prévu pour être beaucoup plus souvent interrogé en lecture que modifié en écriture. La recherche des informations y est donc optimisée pour être rapide et structurée.
- Les données de l'annuaire sont organisées selon un arbre, alors que les bases de données relationnelles gèrent leurs données dans des tables.
- Les données de l'annuaire sont systématiquement protégées. Même en lecture, elles ne sont accessibles qu'après authentification, ce qui n'est pas le cas pour bien des bases de données.

Exercice 8 : protocoles de consultation de boîte aux lettres

Solution

Tableau 9.1 : Comparaison entre POP et IMAP

	POP	IMAP
Lieu de stockage des messages	Chez le client après transfert	Conservés sur le serveur
Espace mémoire sur le serveur	Faible	Grand
Dossiers de courrier	Chez le client	Sur le serveur
Interrogation de la boîte aux lettres de n'importe où	Non	Oui
Possibilité d'utiliser Webmail	Non concerné	Non concerné

Remarque

Un fournisseur d'accès peut avoir intérêt à proposer IMAP à ses clients de messagerie, même s'il en résulte des coûts de stockage importants sur le serveur : il peut proposer de facturer le stockage au-delà de quelques mégaoctets, générant ainsi du chiffre d'affaires.

Exercice 9 : analyse de l'en-tête d'un courrier électronique

Solution

Le message a été émis le 2 mai 2006 à 8 h 27 min 35 s (+ deux heures par rapport à l'heure GMT, ce qui correspond à l'heure d'été en France). Il a été composé à l'aide de l'outil de messagerie de Microsoft Outlook Express 6 avec une priorité normale et a été relayé par plusieurs serveurs de messagerie.

Il faut lire la succession des relais de messagerie à l'envers : l'en-tête contenant l'information sous la forme « reçu de la part de *w* par *x* ; reçu de la part de *x* par *y* ; reçu de la part de *y* par *z* » ; le chemin emprunté est, dans l'ordre chronologique, *z y x w*.

Les trois premiers relais sont des serveurs du fournisseur d'accès Free (avec les adresses IP `62.147.81.254`, `213.228.0.130` et `213.228.0.2`), les deux derniers des serveurs du fournisseur Wanadoo : *mel-rti17.wanadoo.fr* et *ms9.wanadoo.fr* (dont on ne connaît pas l'adresse IP publique).

Remarque

Le protocole ESMTP qui apparaît dans cet en-tête est une extension de SMTP (*Extended SMTP*). La RFC 1651 définit une commande nouvelle, *EHLO* (les lettres *E* et *H* inversées), qui introduit un dialogue ESMTP si les deux parties le reconnaissent. Si le serveur distant ne le reconnaît pas, la commande est ignorée et le dialogue continue en SMTP classique. Si le serveur distant reconnaît *EHLO*, il envoie la liste des extensions qu'il supporte et le client peut alors utiliser celles qu'il souhaite.

Exercice 10 : mise à disposition d'un logiciel par un serveur FTP

Solution

1. L'URL sera *ftp://www.soc.pays /ftp/pub/freeware/logiciel.prog.*
2. Les commandes successives seront :
   - `open nom_du_serveur` pour ouvrir une connexion avec le serveur FTP ;
   - `dir` pour voir la structure de l'arborescence des répertoires ;
   - `cd pub/freeware` pour atteindre le répertoire voulu ;
   - `dir` pour afficher le contenu du répertoire atteint ;
   - `get logiciel.prog` pour prendre le fichier du serveur ;
   - `bye` ou `quit` pour fermer la connexion avec le serveur.

Remarque

Sur les navigateurs récents, le fait que le nom symbolique commence par *www* induit l'utilisation du protocole HTTP. Un serveur Web peut bien évidemment proposer des documents à transférer par d'autres protocoles. Il faut alors explicitement préciser le protocole utilisé (ici FTP).
Quant aux outils FTP modernes, ils proposent une interface graphique qui visualise d'un côté l'architecture des répertoires de la machine de l'utilisateur et de l'autre celle du serveur. Il suffit de cliquer pour se déplacer dans les répertoires du serveur et de sélectionner le fichier voulu, puis de cliquer sur un bouton de transfert (généralement une grosse flèche) pour provoquer le téléchargement du fichier. Inutile d'apprendre les commandes ci-dessus !

Exercice 11 : serveur Web sur un autre port

Solution

Le serveur Web n'est accessible que si les clients connaissent le numéro de port actif. Il faut donc publier celui-ci par exemple avec l'URL *http://www.domaine.pays:8080*, dans laquelle *8080* est le port utilisé (la RFC 1738 standardise cette écriture).

Exercice 12 : serveurs Web et webographie

Solution

L'étudiant 1 fournit une référence technique et précise : le lien qu'il mentionne permet de retrouver directement la publication qu'il a consultée.

L'étudiant 2 a visiblement cherché beaucoup plus loin que ses camarades. Non seulement il donne plus de références, mais celles-ci sont fouillées et correspondent à des sources que l'on peut estimer sérieuses. Il a consulté aussi un blogue et un site où les informations sont en bases de données. L'inconvénient évident de ces derniers liens est leur longueur et le fait qu'il devient impossible de les connaître !

L'étudiant 3 n'a cherché aucune source scientifique. Il s'est très certainement contenté d'un moteur de recherche dans lequel il a tapé « technologie REST ». Les références sont totalement inutiles à qui voudrait travailler sur le sujet : il n'est même pas sûr qu'en allant sur le site du journal gratuit et en indiquant « technologie REST » dans le moteur de recherche de ce site, on trouve l'article qu'aurait consulté l'étudiant 3.

Chapitre 10

Nouvelles applications et sécurité dans les réseaux

Après avoir évoqué les applications les plus répandues dans les réseaux, nous présentons dans un premier temps la ToIP (*Telephony over IP*), qui connaît depuis quelques années un développement spectaculaire, aussi bien dans les entreprises que chez les particuliers. La convergence de l'informatique et des télécommunications a conduit à l'élaboration de nombreuses normes pour adapter le fonctionnement d'Internet aux contraintes temps réel de la téléphonie.

Dans un deuxième temps, nous abordons différents aspects liés à la sécurité et nous traitons le cas particulier des usagers nomades. Nous décrivons le vocabulaire des services et des mécanismes de sécurité défini par l'ISO : authentification, intégrité, non-répudiation… Nous citons quelques exemples de solutions retenues pour parer différents risques et menaces. Enfin, nous abordons le chiffrement, la signature numérique, les certificats, les réseaux privés virtuels, les pare-feu, etc. La panoplie des protections, très vaste, s'accroît avec la créativité des attaquants.

Problèmes et exercices

Exercice 1 : quelques difficultés inhérentes à la ToIP

Solution

1. Deux causes principales à ces désagréments : la perte de plusieurs paquets voix et/ou la gigue qui a dépassé les tolérances admises. En effet, une gigue trop importante équivaut à une perte de paquets puisque les échantillons sont arrivés après l'instant où ils devaient être exploités : ils sont inutilisables.
2. Le flux de signalisation et le flux média n'empruntent pas le même chemin : le plus souvent, la signalisation transite *via* l'IPBX alors que la voix circule directement entre les terminaux. Une erreur de routage IP ou le filtrage intempestif d'un pare-feu ont pu bloquer le flux média. Cette éventualité est plus fréquente pour les communications qui traversent plusieurs équipements (routeurs ou pare-feu), même si les communications à l'intérieur d'un même site ne sont pas à l'abri de ce genre d'incident.

Remarque

Cela illustre bien l'indépendance, héritée de la téléphonie traditionnelle, des canaux signalisation et voix, existant en H.323 comme en SIP. L'utilisation des ports dynamiques dans le protocole RTP pour transporter la voix favorise également ces incidents. Cela montre aussi la nécessité de bien paramétrer les routeurs et les pare-feu, et surtout d'effectuer des tests d'appels dans le même site et entre sites lors de la mise en service de la ToIP.

Exercice 2 : débit réel d'un codec

Solution

1. À 64 kbit/s, on transmet un échantillon de 8 bits toutes les 125 µs, donc 160 échantillons de 8 bits sont émis en 20 ms.
2. 50 trames par seconde, puisqu'une trame prend 20 ms.
3. L'*overhead* lié aux en-têtes des différents protocoles est de : $59 \times 8 = 472$ bits par trame, soit 23 600 bit/s. Pour un codec G.711, le débit réel est : 64 000 + 23 600 = 87 600 bit/s soit un débit réel de : 87,6 kbit/s.
4. Comme on utilise le même moyen pour transporter la parole, l'overhead pour le codec G.729A est identique au précédent, mais le débit utile est beaucoup plus faible : 8 000 + 23 600 = 31 600 bit/s, soit un débit réel de : 31,6 kbit/s.

Remarques

1. Les calculs précédents servent à déterminer la bande passante à consacrer au trafic voix dans un réseau.
2. Les flux supportant les communications téléphoniques sont symétriques, contrairement au trafic données entre un serveur et un client : le flux du client ou flux montant est beaucoup plus faible que le flux du serveur ou flux descendant.

Exercice 3 : communication entre terminaux SIP

Solution

1. Oui, sinon les nouveaux terminaux ne pourraient pas communiquer avec les anciens.
2. Puisque la voix est transmise, l'appelé a décroché son combiné. On devrait donc trouver la réponse positive `200 OK` après *Ringing*. Or, l'IPBX continue d'envoyer des *Ringing* !
3. Le terminal SIP raccroche le premier. Pour signaler qu'il a raccroché, il envoie le message *CANCEL* au lieu de *BYE* car il n'a pas reçu le `200 OK` attendu.
4. Aucun dysfonctionnement n'est constaté dans le comportement entre les anciens terminaux SIP et l'IPBX et entre les différents types de terminaux SIP. Les problèmes ne proviennent pas du réseau puisque les communications sont possibles, et la trace réseau a montré que *Invite* de l'IPBX est correctement transmis mais ignoré du terminal. Il s'agit sans doute d'un problème protocolaire (compatibilité des messages SIP entre l'IPBX et les nouveaux terminaux SIP). Il faut sûrement modifier le logiciel des nouveaux terminaux SIP ou demander au constructeur de l'IPBX s'il a effectué des tests d'interopérabilité avec ces terminaux. L'administrateur est impuissant dans cette situation.

Remarque

Cet exemple inspiré d'un cas réel montre les limites de l'interopérabilité « native » promise par SIP !

Exercice 4 : code de César

Solution

Le message écrit est : `LesmetiersdInternet` (Les métiers d'Internet).

Le code consiste en un décalage de 14 lettres, ce qui donne :

- clair : `abcd`**`e`**`fghijklmnopqrstuvwxyz` ;
- chiffré :`opqr`**`s`**`tuvwxyzabcdefghijklmn`.

En français, la lettre la plus fréquente est le *e* : ici il y a 5 `s` et 3 `h`. Il est logique de tester en priorité `s = e`. Le reste vient tout seul car le décalage est constant. Le système est très facilement cassable dès qu'on connaît les fréquences des lettres dans la langue utilisée !

Exercice 5 : cryptanalyse

Solution

1. Il y a 2^{64} clés possibles. En moyenne, vous en essaierez la moitié (une seule avec beaucoup de chance et toutes avec beaucoup de malchance, ce qui fait en moyenne la moitié), soit 2^{63}.
2. Une picoseconde = 10^{-12} s. Le temps moyen nécessaire est donc de : 263×10^{-12} s.

   En utilisant $10^3 = 2^{10}$, on obtient $2^{63} \times 10^{-12}$ s = 2^{23} s = 8 388 608 s = 2 330 h = 97 jours = 3 mois.
3. Vous améliorez vos « performances » avec une puissance de calcul plus grande (1 000 fois plus grande par exemple ; le temps moyen devient deux heures). Deux solutions pour vous rendre la tâche impossible : augmenter la longueur de la clé (avec une clé de 128 bits et la puissance de calcul 1 million de fois plus grande qu'à la question précédente, il vous faudra tout de même 6 milliards de milliards de siècles… en moyenne !) et changer la clé régulièrement (si on garde les valeurs de la première question avec un temps moyen de trois mois, il est judicieux de changer la clé toutes les semaines !).

Exercice 6 : cassez un système !

Solution

Dans un code de substitution, la fréquence des lettres codées est la même que la fréquence des lettres dans la langue utilisée. Si vous savez que la clé est de longueur 5, il faut découper le message en blocs de cinq caractères :

- `KAZUI VZYTJ ZXFPD IFFJC ZQXWQ ZXQHR JYRHC OEKXI JZXLB VSNQT MQSYD TMSWJ IHTOS CUWRC YQQOT NCZHA VGYRB IQALT IFIDG MUAHG.`

Il faut ensuite traiter tous les premiers caractères de blocs, tous les deuxièmes, etc. Vous obtenez, par exemple, pour tous les premiers caractères : `KVZIZZJOJVMTICYNVIIM`.

Ces données font apparaître quatre *I* et trois *Z*. On peut penser que l'une des deux lettres représente le *E*, lettre la plus fréquente en français.

Il faut faire de même avec les deuxièmes caractères : `AZXFQXYEZSQMHUWQCGQFU`.

Ces données font apparaître quatre *Q* et jamais plus de deux fois une autre lettre. On peut raisonnablement penser que le *Q* code le *E*.

Les troisièmes lettres donnent : `zyffxqrkxnsstwqzyaia`.

Ici, rien de significatif : deux *F*, deux *Q*, deux *S*, deux *A*, deux *Z*…

Les quatrièmes lettres donnent : `UTPJWHHXLQYWOROHRLDH`.

Ces données font apparaître quatre *H* et jamais plus de deux fois une autre lettre. Il y a de bonnes chances pour que le *H* code le *E*.

Les cinquièmes lettres donnent : `IJDCQRCIBTDJSCTABTGG`.

Ici encore rien de vraiment significatif : trois *C*, deux *I*, deux *J*, deux *B*, deux *G*…

Utilisons l'hypothèse la plus vraisemblable : le *E* est codé par *Q* en tant que troisième lettre et *H* en tant que quatrième.

Le message devient alors, en décodant toutes les deuxièmes et quatrièmes lettres :

- `.O.R. .N.Q. .L.M. .T.G. .E.T. .L.E. .M.E. .P.U. .N.I. .G.N.` ;
- `.E.V. .A.T. .V.L. .I.O. .E.L. .Q.E. .U O. .E.I. .T.A. .I.E.`.

Si cette hypothèse est la bonne, on peut remarquer que la lettre *Q* est presque toujours suivie d'un *U* en français. Nous obtenons alors le décodage de la cinquième lettre et de la troisième :

- `.OURT .NTQU .LAMO .TAGN .ESTB .LLEC .MMEN .PEUT .NSIM .GINE .ENVO .ANTU .VOLD .IRON .ELLE .QUEL .UTOM .EVIE .TDAR .IVER.`

Quelques essais pour la première lettre montrent que le *E* est codé par *Z*. On décode finalement :

`POURT ANTQU ELAMO NTAGN EESTB ELLEC OMMEN TPEUT ONSIM AGINE RENVO YANTU NVOLD HIRON DELLE SQUEL AUTOM NEVIE NTDAR RIVER`, soit, en mettant les espaces et les apostrophes, le refrain d'une chanson de Jean Ferrat :

- `POURTANT QUE LA MONTAGNE EST BELLE COMMENT PEUT-ON S'IMAGINER EN VOYANT UN VOL D'HIRONDELLES QUE L'AUTOMNE VIENT D'ARRIVER.`

Exercice 7 : *The man in the middle*

Solution

1. Le mécanisme Diffie-Hellman est très intéressant car *A* et *B* vont partager un secret (g^{xy} mod n) alors que celui-ci n'a pas été transmis. *A* et *B* ont chacun choisi de leur côté un nombre (x ou y) et l'ont transmis à l'autre sous une forme chiffrée. *A priori*, la connaissance de g et de n ne permet pas de retrouver x ou y.
2. Reprenons l'échange de la question précédente avec *C* au milieu. *A* choisit un nombre x et calcule g^x mod n qu'il envoie à *B*… non ! à *C*, qui intercepte le message. *C* choisit un nombre z et calcule g^z mod n qu'il envoie à *B*. Celui-ci choisit un nombre y et calcule g^y mod n qu'il envoie, croit-il, à *A*. En fait, *C* intercepte le message et envoie g^z mod n à *A*.

   Comme dans le scénario précédent, *A* calcule $[g^z \bmod n]^x \bmod n = g^{xz} \bmod n$, secret qu'il croit partager avec *B* alors qu'il le partage avec *C*. De même, *B* calcule $[g^z \bmod n]^y \bmod n = g^{yz} \bmod n$, secret qu'il croit partager avec *A*… Le tour est joué, *C* partage un secret avec chacun des deux correspondants et peut déchiffrer les communications, voire infiltrer des messages dans la communication ou en détruire.

Remarque

La faiblesse de ce système tient à son absence d'authentification (voir exercice 9).

Exercice 8 : services de sécurité

Solution

Le premier message est chiffré avec la clé publique de Bob ; il y a donc confidentialité. Son contenu est chiffré avec la clé privée d'Alice ; seule Alice dispose de cette clé : elle fournit ainsi une authentification et une preuve de non-répudiation. Le contenu déchiffré fait apparaître des données M, un condensé par une fonction de hachage $H(M)$; il est donc possible de vérifier la cohérence de l'un avec l'autre (contrôle d'intégrité). Enfin, ce message transporte une clé unique qu'Alice propose à Bob pour la suite de leurs échanges dans un système symétrique.

Le deuxième message ne peut pas avoir été envoyé par Alice puisqu'il utilise la clé privée de Bob.

Le troisième message est correct du point de vue de l'usage des clés, mais il est totalement inutile puisqu'il véhicule une clé publique que l'on trouve dans un annuaire, par exemple.

Exercice 9 : authentification

Solution

1. *A* et *B* se sont mutuellement authentifiés après le « défi » qu'ils se sont lancé : êtes-vous capable de chiffrer avec notre clé partagée le nombre aléatoire que je viens de vous envoyer ?
2. L'attaquant *C* peut encore faire des ravages ! Imaginons qu'il intercepte le premier message. Il le change et envoie à *B* l'identité de *A* accompagnée d'un nombre aléatoire N_C qu'il a lui-même choisi. *B* envoie en retour un nombre aléatoire N_B et le nombre envoyé par *C*, chiffré avec la clé partagée K_{AB}. Il n'a que faire de ce dernier mais le nombre N_B est très précieux. Un peu plus tard, se faisant toujours passer pour *A*, *C* envoie à *B* le message initial : son identité (celle de *A*…) et le nombre N_A intercepté au début. *B* peut raisonnablement penser qu'il s'agit du début d'une nouvelle procédure d'authentification : il envoie en retour un nombre aléatoire N'_B et le nombre envoyé par *C* (c'est-à-dire N_A), chiffré avec la clé partagée K_{AB}. Et le tour est joué. *C* est maintenant en possession de N_B en clair et de N_A chiffré avec la clé partagée K_{AB}. C'est le message que *B* aurait dû envoyer à *A* lors de la requête initiale. *C* l'envoie donc, et *A* pense alors qu'il discute avec *B*…

Remarque

Le scénario est devenu complexe, il faut de plus imaginer que *B* accepte deux sessions différentes avec *A*… Plus la protection est grande et plus l'attaquant doit faire preuve d'ingéniosité ! Une solution à ce nouveau problème pourrait être de dater les messages et de contraindre l'intervalle de temps pour la réponse, ce qui peut cependant gêner les processus normaux autorisés !

Exercice 10 : règles d'un pare-feu

Solution

1. L'usurpation d'adresse correspond à des messages provenant de l'extérieur du réseau dont l'adresse d'émetteur est une adresse interne. Notons : `ouraddr = 195.45.3.0/24` (toutes les adresses de notre réseau) et `anyaddr` = n'importe quelle adresse. Les règles du pare-feu sont :
   - effacer toutes les règles en entrée ;
   - refuser les messages en entrée dont l'adresse source est `ouraddr`, l'adresse destination est `ouraddr`, le protocole est TCP et le bit SYN est mis à 1 ;
   - refuser les messages en entrée dont l'adresse source est `ouraddr`, l'adresse destination est `ouraddr` et le protocole est ICMP.
2. *iptables -A INPUT -p tcp -m multiport --destination-port 21,23 -s ! 127.0.0.1 -j DROP* : fermeture des ports des serveurs FTP et de Telnet (le mot de passe circule en clair sur le réseau) [ces deux applications doivent être remplacées par SFTP et SSH]. Avec cette règle, les serveurs FTP et Telnet ne répondent plus, sauf pour les tests en boucle locale (adresse `127.0.0.1`).

   iptables -A INPUT -p tcp --syn -m limit --limit 10/s -j ACCEPT : limitation du nombre de requêtes d'ouverture de connexion (SYN) à 10 par seconde pour empêcher l'attaque *SYN flooding*.

   iptables -A INPUT -p tcp -m mac --mac-source 00:02:B3:98:41:08 –j DROP : interdiction de toutes les connexions TCP en provenance de la machine dont l'adresse MAC est `00:02:B3:98:41:08`. Tous les segments TCP sont rejetés.
3. *iptables* applique la stratégie de « *first fit and not best fit* ». Il fait un traitement séquentiel des règles en examinant la condition (si condition vérifiée, alors faire quelque chose). La première règle qui satisfait à la condition de filtrage est appliquée au paquet.

Exercice 11 : détection d'intrusion

Solution

alert icmp any any -> any any (ttl = 255)
alert tcp any any -> 192.168.100.0/24 any (flag=URG, PSH, FIN)
alert udp any any -> 192.168.100.0/24 31337 ()
alert tcp any any -> any any (flag = ACK ; num_ack = 0)

Exercice 12 : paradoxe des anniversaires

Solution

1. Pour $n = 2$, cette probabilité est $p = (1–1/365) = 364/365 = 0{,}99$ (il suffit que le second ne soit pas né le même jour que le premier ; il reste donc 364 jours possibles) ; pour $n = 3$, on obtient $p = (1–1/365) \times (1–2/365) = (364/365) \times (363/365) = 0{,}97$ (il suffit d'ajouter que le troisième n'est né ni le même jour que le premier ni que le deuxième ; il reste donc 363 jours possibles pour lui).

   Pour une valeur $n \geq 4$, on a $p = (1–1/365) \times (1–2/365) \times \ldots \times ((1–(n–1)/365)$.

   L'application numérique montre qu'il suffit d'avoir $n \geq 23$ pour que p devienne inférieure à un sur deux : pour $n = 22$, on a $p = 0{,}52$ et $n = 23$, $p = 0{,}49$. Cette propriété est nommée le *paradoxe des anniversaires*. En effet, il paraît peu probable au commun des mortels de trouver, dans une classe de lycéens par exemple, deux jeunes qui fêteront leur anniversaire le même jour ! Et pourtant, le calcul montre que l'on a plus d'une chance sur deux que cela se produise dès que le groupe comprend au moins 23 personnes.
2. On peut appliquer le même raisonnement à des messages et à leurs signatures. Si la signature est de longueur n, il y a 2^n signatures possibles. La probabilité que k messages aient des signatures différentes est $p = (1–1/2^n) \times (1–2/2^n) \times \ldots \times ((1–(k–1)/2^n)$. On peut comme à la question 1 chercher quand cette probabilité devient inférieure à un sur deux (voir tableau 10.1).

Tableau 10.1 : Taille de la signature et nombre de messages à générer

Taille de la signature n (bits)	Nombre de messages à générer k
8	13
16	213
32	54 562
64	$9{,}6\ 10^9$
128	$1{,}5\ 10^{19}$
160	10^{24}
256	$2{,}8\ 10^{38}$

   L'application à la sécurité est simple : si une signature fait 8 bits, il suffit de générer 13 messages pour que la probabilité que deux d'entre eux aient la même signature soit supérieure à un sur deux. Avec une signature de 16 bits, il en faut 213. Avec des signatures plus longues, le nombre de messages à générer est dissuasif.

Exercice 13 : IPSec et NAT

Solution

1. Les routeurs décrémentent le champ *TTL* de l'en-tête de chaque paquet et refont le calcul du bloc de contrôle d'erreur de l'en-tête. Ces deux champs ne doivent donc pas entrer dans le mécanisme d'authentification ; sinon, tous les paquets IP seraient falsifiés à la première traversée d'un routeur.
2. Le mécanisme d'authentification garantit les adresses IP, il ne garantit pas les personnes qui les utilisent…
3. Lorsqu'un paquet traverse un routeur pare-feu qui utilise le mécanisme NAT, l'adresse IP interne est remplacée par une nouvelle adresse IP publique. Cela est incompatible avec le mécanisme NAT.

Remarque

Il a fallu trouver des adaptations de NAT pour contourner le problème. C'est l'une des occasions qui nous a fait qualifier de « bricolage » bien des solutions proposées.

Exercice 14 : sécurité d'un WLAN

Solution

La sécurité d'un réseau sans fil proposée par les standards 802.11 s'appuie sur : l'identification du WLAN, l'enregistrement des adresses MAC des participants et le chiffrement. Cette sécurité est notoirement insuffisante car elle peut se contourner : l'identifiant est statique et s'affiche en clair dans les outils de configuration ; l'adresse MAC peut se changer dans une carte réseau ; quant au chiffrement standard, il utilise une clé WEP, mécanisme d'authentification simple qui fait appel à un algorithme de chiffrement cassable… Le standard 802.11i ou encore WPA (*Wi-Fi Protected Access*) offrent une meilleure sécurité avec des protocoles d'authentification plus évolués.

WPA est un sous-ensemble du standard 802.11i. Il utilise EAP (*Extensible Authentification Protocol*) et TKIP (*Temporal Key Integrity Protocol*). EAP, décrit dans le standard 802.11x, sert à authentifier les équipements du réseau, et TKIP renforce la sécurité du protocole WEP. Par ailleurs, 802.11i utilise un algorithme de chiffrement plus robuste que celui de la clé WEP.

Exercice 15 : choix d'un VPN

Solution

Dans cette agence, il s'agit de différencier le traitement de certaines communications par rapport à d'autres. Une solution de niveau opérateur (MPLS) ou IP avec AH ne peut donc pas convenir (AH ne procurant pas de confidentialité). La solution IP en mode tunnel avec ESP est à écarter également car elle ne convient pas pour les communications mobiles.

Les solutions inadaptées sont IPSec en mode transport avec AH, IPSec en mode tunnel avec AH, IPSec en mode tunnel avec ESP et MPLS.

Exercice 16 : peer-to-peer et sécurité

Solution

1. Le pare-feu de l'entreprise n'est pas bien configuré : l'administrateur aurait pu interdire le téléchargement peer-to-peer. De plus, le poste de l'utilisateur possède un antivirus non performant ou pas à jour ; ce dernier aurait dû détecter le virus.
2. PGP n'apporte aucune solution au problème puisque c'est une méthode de chiffrement des courriers électroniques et qu'elle ne possède aucun moyen de lutte contre les virus.