Notes sur les réseaux

L. Darré, C. Dürr (IUT d'Orsay)

30 janvier 2004

Table des matières

Chapitre 1

Les protocoles

1.1 Présentation

Ich will wissen was die Welt
im Innersten zusammen hält

(Je veux savoir comment Internet fonctionne)

— Johann Wolfgang von Goethe dans Faust

1.1.1 Introduction

Le but de ce cours est d'une part de vous donner assez de culture générale sur les réseaux pour que vous puissiez comprendre tout ce qui est mis en oeuvre à tous les niveaux lorsque vous cliquez dans un hyperlien dans une page web par exemple, et d'autre part de vous permettre de faire de l'administration réseau. Certaines sections de ce document vont plus en détail que le cours, et vous sont données à titre d'information. Si vous y trouvez des fautes, faites le savoir, vous recevrez du nougat viêtnamien en retour.

La série des protocoles TCP/IP est utilisée de nos jours pour relier des machines d'architectures différentes utilisant des systèmes d'exploitation différents. Ces protocoles, dont les sources sont disponibles gratuitement, sont devenus, avec la constitution de l'Internet, réseau mondial qui relie plusieurs millions d'ordinateurs, les protocoles de communication les plus répandus.

1.1.2 Modèle OSI

Les protocoles réseau sont organisés en couches pour permettre à des systèmes différents de communiquer de manière transparente. Ceci est tout à fait similaire à l'organisation en couches des machines : Jeu d'instruction du processeur, système d'exploitation, langage de programmation, bibliothèque d'interface graphique. Elle permet d'exécuter le même programme sur un 486 ou sur un Pentium III, où alors de recompiler tel quel un programme pour une autre architecture, ou de changer d'interface graphique sans changer le reste du programme.

Les protocoles réseau sont généralement organisés en couches à l'image du modèle OSI (Open System Interconnection) fourni par l'ISO et décrit par H. Zimmerman du CNET. Ce modèle sert de référence mais n'a jamais été implémenté tel quel entièrement. Ce modèle possède sept couches, les quatre premières étant appelées les couches basses, les trois autres, les couches hautes :

1.1.2.1 La couche physique

La première couche de l'architecture OSI a pour objectif de conduire les éléments binaires jusqu'à leur destination sur le support physique. Le medium physique est par exemple :

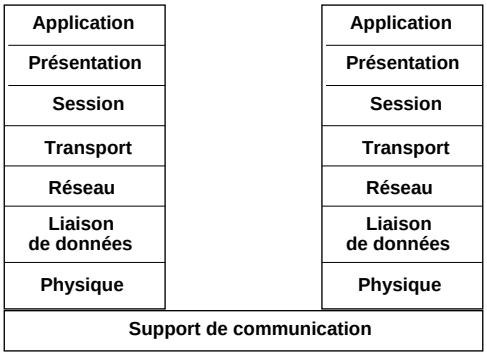

FIG. 1.1 – L'architecture OSI

- la paire torsadée : formée de deux à huit brins de cuivre agencés en spirale. Le câble en paires torsadées catégorie 5 est agréé pour faire passer du 100 Mbps (Mega bits par seconde) sur 100 mètres. Ce support est largement utilisé pour les réseaux locaux Ethernet et Token-ring et pour les réseaux téléphoniques. Il est torsadé pour être résistant aux ondes électromagnétiques environantes. La catégorie spécifie le nombre de torsations par centimètre.
- Le câble coaxial : composé de conducteurs cylindriques de même axe séparés par un isolant. Ce cable est aussi utilisé pour relier une télévision à son antenne. Il est rarement utilisé aujourd'hui.
- La fibre optique : constituée d'un cylindre de verre très fin entouré d'une couche concentrique de verre. C'est un support idéal qui est insensible à l'environnement et qui permet des débits de plusieurs Gbps (Giga bits par seconde).
- Les faisceaux hertziens

On caractérise la qualité du support physique et de son protocole de communication de niveau physique par un taux d'erreurs bit ou BER (Bit Error Rate) qui indique, en moyenne, le nombre de bits envoyés pour un bit en erreur.

1.1.2.2 La couche liaison

La couche liaison fiabilise la précédente en corrigeant les erreurs qui ont pu se produire afin que le taux d'erreurs résiduelles soit négligeable. Le niveau 2 doit donc posséder des fonctionnalités lui permettant de détecter les erreurs à l'arrivée d'un bloc d'informations et de les récupérer. Elle comprend aussi les règles nécessaires pour partager un support physique unique entre plusieurs stations.
L'ISO fournit des normalisations dans le domaine des réseaux locaux pour les méthodes d'accès et les protocoles de liaison, comme par exemple :

- ISO 8802.3 pour la technique d'accès CSMA/CD (Ethernet)
- ISO 8802.4 pour la technique d'accès d'un jeton sur un bus
- ISO 8802.5 pour la technique d'accès d'un jeton sur un boucle

1.1.2.3 La couche réseau

La couche réseau doit permettre d'acheminer correctement les paquets d'informations jusqu'à l'utilisateur final. Ce niveau a trois fonctions principales :

1. L'adressage : la gestion de l'adressage consiste à ajouter des adresses complètes dans les différents paquets pour qu'ils atteignent leur destinataire. Les adresses forment un ensemble très vaste qui regroupe toutes les machines terminales du réseau. Les machines non-terminales sont des composants du réseau qui ne peuvent être adressés à ce niveau.

2. Le routage : il permet d'acheminer les paquets d'informations vers leur destination au travers du maillage des noeuds de commutation.
3. Le contrôle de congestion : il doit éviter les embouteillages de paquets dans le réseau.

Le développement des fonctionnalités de niveau réseau peut être envisagé avec deux modes de connexion.

1. le mode orienté connexion (également appelé circuit virtuel) : l'émetteur et le récepteur se mettent d'accord sur un comportement commun et négocient les paramètres. Un chemin est alors établi pour tous les paquets transitant entre les deux machines. L'arrivée est sure et les paquets sont reçus dans l'ordre dans lequel ils ont été émis. Une communication dans le mode connecté est donc constituée de trois phases :
   (a) établissement de la connexion
   (b) transfert des données
   (c) libération de la connexion

   Par exemple, ISO8208 ou CCITT X.25 définissent le protocole de réseau en mode connecté; ce protocole est le plus souvent appelé X25.
2. Le mode sans connexion (également appelé datagrammes) : l'émetteur n'impose aucune contrainte au récepteur. Il informe le réseau qu'il veut communiquer avec une machine dont l'adresse est incluse dans le paquet. Le réseau prend alors le relais et s'occupe de tout. Ce mode est dit non fiable car il ne garantit pas l'arrivée. Chaque paquet est routé indépendamment des autres.
   L'ISO 8473 définit le protocole de réseau en mode non connecté connu sous le nom d'Internet ISO. C'est une normalisation du protocole IP.

1.1.2.4 La couche transport

C'est la couche charnière entre la couche haute et la couche basse. C'est l'ultime niveau qui s'occupe de l'acheminement de l'information. Elle est la pour compléter ce qui a été fait par les couches précédentes. Elle doit permettre de donner à l'utilisateur la qualité de service de transmission d'informations susceptible de le satisfaire. Le protocole de niveau 4 à mettre en oeuvre va fortement dépendre du service rendu par les trois premières couches et de la demande de l'utilisateur.

1.1.2.5 La couche session

Elle possède les fonctionnalités nécessaires à l'ouverture, à la fermeture et au maintien de la connexion.

1.1.2.6 La couche présentation

Elle se charge de la syntaxe des informations échangées au niveau application. Les données sont structurées de façon à ce que les diverses applications puissent s'entendre sur la représentation des données.

1.1.2.7 La couche application

Cette dernière couche contient toutes les fonctions impliquant des communications entre système, en particulier si elles ne sont pas réalisées par les couches inférieures. Elle s'occupe essentiellement de la sémantique.

1.1.3 La couche de liens

Le but de la couche de liens de la série de protocole TCP/IP est d'envoyer et de recevoir :
- des datagrammes IP pour le module IP.
- des requêtes ARP et des réponses pour le module ARP.
- des requêtes RARP et des réponses pour le module RARP.

TCP/IP supporte beaucoup de couches de liens différentes selon le type de matériel et de réseau utilisé (Ethernet, Token ring, FDDI, lignes séries ...).

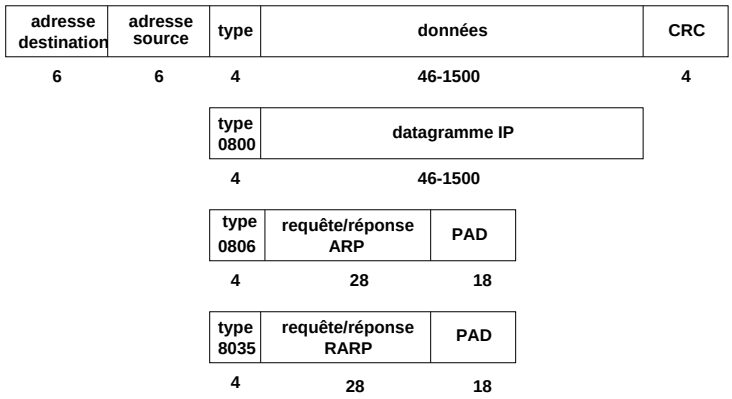

FIG. 1.2 – L'encapsulation Ethernet

1.1.3.1 Ethernet et l'encapsulation IEEE802

Ethernet fait référence à un standard publié en 1982 qui est prédominant dans la technologie des réseaux locaux utilisé avec TCP/IP. Ethernet utilise une méthode d'accès appelée CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Il opère à 10 ou 100 Mbps (maintenant à 1 Gbps) et emploie des adresses sur 48 bits. IEEE (Institute of Eletrical and Electronic Engineers) publiera plus tard le standard 802.3.

L'encapsulation des datagrammes IP dans les réseaux Ethernet (voir la figure 1.2) est définie, entre autre, dans le RFC (Request For Comment) 894. Le format de la trame utilise des adresses sources et destination d'une taille de 48 bits (6 octets). Les adresses sont appelées adresses matérielles. Les protocoles ARP et RARP effectuent les correspondances entre les adresses IP sur 32 bits et les adresses matérielles sur 48 bits.

Le champ *type* indique le type de données qui suivent. Le champ *CRC* (Cyclic Redundancy Check) est une somme de contrôles qui détecte les erreurs dans le reste de la trame. Il est aussi appelé FCS (Frame Check Sequence). Les trames doivent avoir une taille minimum, c'est-à-dire que la partie donnée doit comprendre au moins 46 octets et une taille maximum ou MTU (Maximum Transmit Unit) de 1500 octets.

1.1.4 Architectures des réseaux IP

Les principaux protocoles définis dans l'architecture TCP/IP sont les suivants :
- IP (Internet Protocol) est un protocole de niveau réseau assurant un service sans connexion.
- ICMP (Internet Control Message Protocol) est un protocole d'envoi de messages entre couches IP de différents noeuds du réseau (cf 1.2.3).
- IGMP (Internet Managment Group Protocol) est un protocole de gestion d'appartenance à un groupe multicast.
- TCP (Transport Control Protocol) est un protocole de niveau transport qui fournit un service fiable avec connexion.
- FTP (File Transfert protocol) est un protocole de transfert de fichiers.
- SMTP (Simple Mail Transfert Protocol) est un protocole de messagerie électronique.
- TELNET Protocol est un protocole de présentation d'écran.

Ces protocoles se présentent sous la forme d'une architecture en couches qui inclut également, sans qu'elle soit définie, une interface d'accès au réseau.

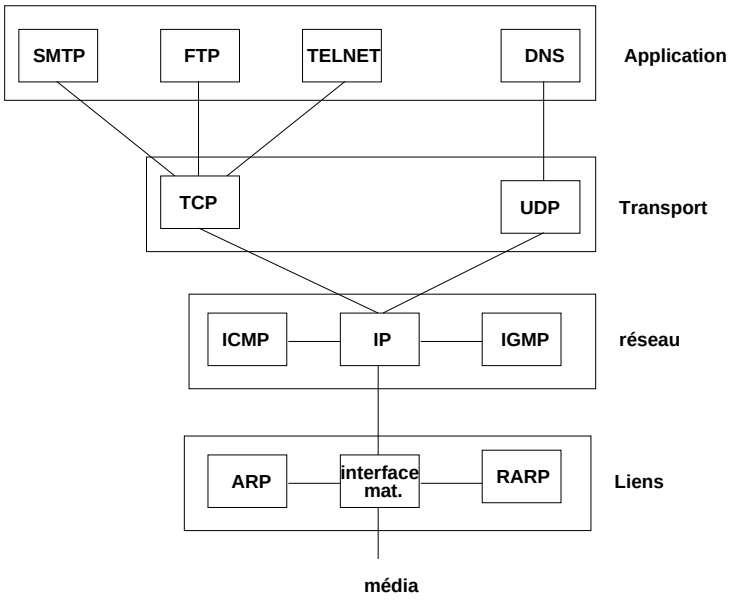

FIG. 1.3 – Divers protocoles de la série de protocoles TCP/IP

Cette architecture est basée sur le protocole IP qui correspond au niveau 3 de l'architecture du modèle de référence. IP a pour but de transporter les paquets, appelés datagrammes, d'une extrémité à l'autre du réseau. Les paquets sont indépendants les uns des autres et sont routés individuellement dans le réseau par chaque commutateur. La sécurisation apportée par ce protocole est très faible : pas de détection de paquets perdus ou de possibilités de reprise sur erreur. Le protocole TCP regroupe les fonctionnalités du niveau 4 du modèle de référence. C'est un protocole assez complexe qui possède de nombreuses options permettant de résoudre tous les problèmes de perte des niveaux inférieurs. TCP fonctionne en mode connecté, contrairement à UDP, deuxième protocole disponible dans cette architecture, qui fonctionne dans un mode sans connexion et sans beaucoup de fonctionnalités. Les protocoles au dessus de TCP ou UDP sont de type applicatif et proviennent en grande partie du monde Unix. La puissance de cette architecture provient de sa capacité à évoluer au dessus de tous les réseaux existants. Pour passer d'un réseau quelconque à un autre (local ou étendu, à commutation de cellules ou par paquets), un équipement spécifique, appelé routeur, va acheminer les paquets IP entre les machines sur lesquelles le protocole IP est aussi implémenté (cf 1.4).

Le service rendu par ce réseau de réseaux est du type 'best effort', ce qui signifie que le réseau fera de son mieux pour écouler le trafic.

Les applications disponibles au dessus de TCP/IP sont nombreuses (SMTP, FTP, WWW ...).

1.2 Couche réseau : IP

1.2.1 Internet Protocol

1.2.1.1 Introduction

IP est le "cheval de trait" de la série des protocoles TCP/IP. L'ensemble des données de TCP, UDP, ICMP et IGMP sont transmises en tant que datagrammes IP. IP procure un service :

– non fiable : il n'existe aucune garantie pour que le datagramme IP arrive avec succès à sa destination ; On dit que IP fournit un service de moindre effort. Lorsqu'un problème survient, IP se contente de

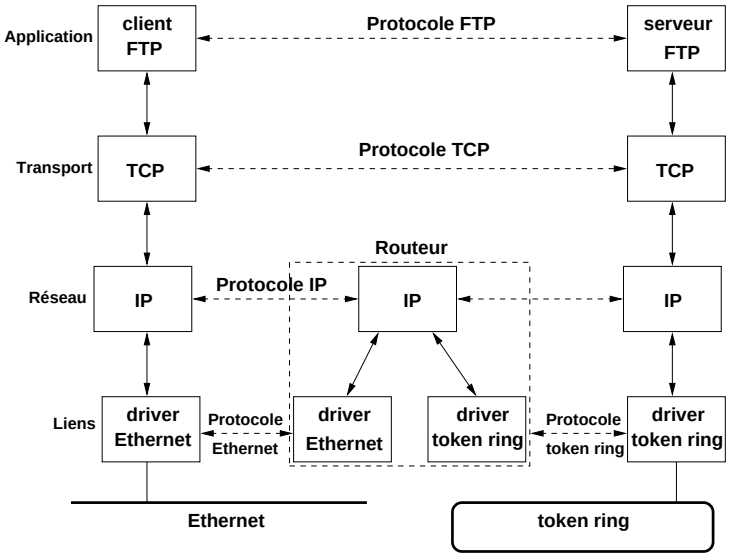

FIG. 1.4 – Deux réseaux connectés par un routeur

rejeter le datagramme et d'essayer d'envoyer un message ICMP à la source. La fiabilité doit être assurée par les couches supérieures.

– sans connexion : les datagrammes sont gérés indépendamment les uns des autres. Deux datagrammes consécutifs émis par une même source au même destinataire peuvent emprunter un chemin différent et arriver dans le désordre.

1.2.1.2 En-tête IP

La taille d'un en-tête est de 20 octets (sans les options). Sa structure est la suivante :

– La *version* courante d'IP est la version 4 d'où son appellation IPv4.

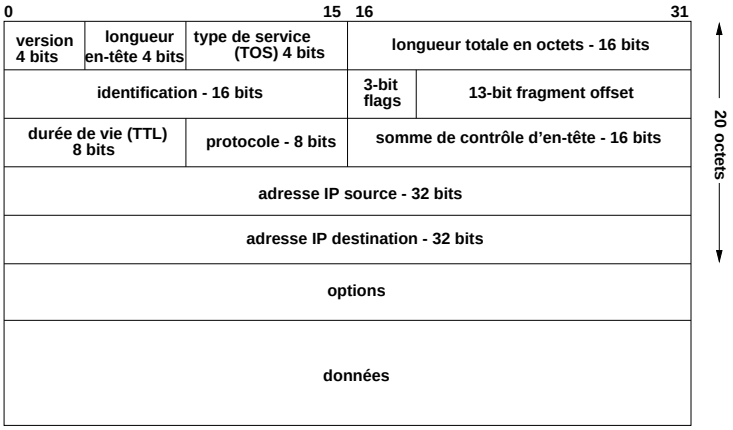

FIG. 1.5 – Format d'un datagramme IP

- La *longueur d'un en-tête* est le nombre de mots de 32 bits figurant dans l'en-tête. Il est donc égal à 5 s'il n'y a pas d'options et à 15 (1111) au maximum.
- Le champs *type de service* est composé d'un champs de 3 bits qui est ignoré, de 4 bits de TOS et d'un bit inutilisé (mis à 0). Les 4 bits de TOS représentent :
  - délai minimum
  - débit maximum
  - fiabilité maximum
  - coût minimum

  Seul l'un de ces 4 bits peut être positionné à 1. Les 4 bits à 0 signifient un service normal. Par exemple, les applications telnet et rlogin qui s'utilisent de façon interactive, ont le bit "délai minimum" à 1 puisqu'elles assurent un dialogue avec un être humain où on veut minimiser le temps de réponse. Par contre, le transfert de fichiers par FTP requiert un débit maximum et a donc le bit correspondant positionné à 1.
- le champ *longueur totale* contient la taille totale en octets du datagramme IP en-tête comprise. La taille maximale d'un datagramme IP, compte-tenu des 16 bits de ce champs, est de 65535 octets. En pratique, de nombreuses applications limitent d'elles-mêmes la taille des données utilisateur.
- Le champ *time-to-live* (TTL) donne une limite supérieure au nombre de routeurs qu'un datagramme peut traverser, ce qui limite la durée de vie du datagramme. Le TTL est décrémenté de 1 à chaque traversée d'un routeur et le datagramme est rejeté si le champ atteint 0. L'expéditeur est alors prévenu par un message ICMP. Ce mécanisme empêche les paquets de rester éternellement dans des boucles de routage.
- Le champ protocole identifie le protocole ayant fourni à IP les données à envoyer (par exemple TCP). Consultez */etc/protocols* pour une liste de codes de protocoles.
- La *somme de contrôle d'en-tête* est calculée à partir du contenu de l'en-tête. Cette somme est calculée par le destinataire et si une erreur est détectée, le datagramme est rejeté.
- Chaque datagramme IP contient l'*adresse IP source* et l'*adresse IP destination*.
- Le champ *options* est rarement utilisé. Il sert, par exemple, pour l'enregistrement de la route.

1.2.1.3 Adressage

Chaque interface sur le réseau Internet possède une adresse IP unique de 32 bits découpée logiquement en deux champs :

1. l'adresse réseau (bits de poids fort)
2. l'adresse locale (bits de poids faible)

Elle est notée en décimale et tient sur quatre octets de la forme a.b.c.d (exemple : 129.175.14.150). Ce format est appelé notation décimale pointée (dotted-decimal notation).

L'espace des adresses n'est pas structuré (pas d'arborescence par pays, etc) mais découpé en 5 classes. La figure 1.6 montre les différentes classes d'adresse.

L'adresse de réseau est délivrée par l'InterNIC (Internet Network Information Center) au niveau mondial. En France, c'est le NIC-France (anciennement l'INRIA) qui s'occupe de la distribution des réseaux.

- Les classes A, B et C adressent des équipements :

Classe	nombre de réseaux	Gamme	nombre d'adresses par réseau
A	126	0.0.0.0 à 127.255.255.255	$256^3 - 2$
B	64x256	128.0.0.0 à 191.255.255.255	$256^2 - 2$
C	$31x256^2$	192.0.0.0 à 223.255.255.255	254

- La classe D est utilisée pour la diffusion de groupe (multicast)
- La classe E est actuellement réservée

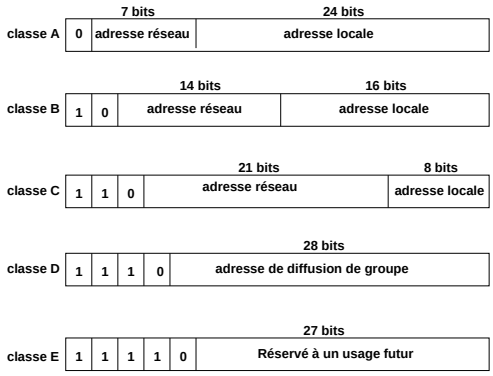

FIG. 1.6 – Les classes d'adresse Internet

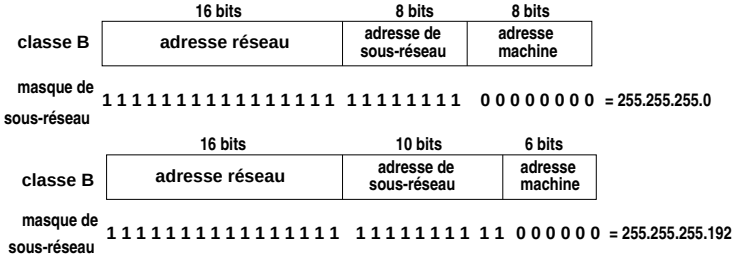

FIG. 1.7 – Exemples de masque de sous-réseau

1.2.1.4 Les sous-réseaux

Les adresses IP étant limitées en nombre, la notion de sous-réseau s'est développé. Elle permet de définir des réseaux physiques multiples, reliés par des routeurs en utilisant une seule adresse réseau. On utilise pour cela un masque de réseau (netmask) qui découpe l'adresse locale en adresse sous-réseau et en adresse machine. Chaque bit du masque de sous-réseau positionné à 1 indique que la position correspondante de l'adresse fait partie de l'adresse réseau, les bits positionnés à 0 indiquant l'adresse locale.

Exemple pour le réseau de classe B 129.175.0.0 de deux masques de sous-réseau, l'un utilisant un identificateur de sous-réseau de 8 bits, l'autre de 10 bits :

Les possibilités d'utilisation d'un sous-réseau de classe C sont assez limitées, l'adresse locale ne comportant que 8 bits. A partir de sa propre adresse IP (ex 129.175.5.53) et de son masque de sous-réseau (ex 255.255.255.0) , une machine peut déterminer si un datagramme IP est destiné à :

- une machine sur son propre sous-réseau (ex 129.175.5.78)
- une machine sur un autre sous-réseau de son propre réseau (ex 129.175.19.120)
- une machine sur un réseau différent (ex 195.120.34.23)

Un réseau, constitué de plusieurs sous-réseaux définis à l'aide d'un masque de sous-réseau, doit satisfaire les règles suivantes :

- le masque de sous-réseau doit être identique pour tous les sous-réseaux dérivés d'un même numéro de réseau

– les sous-réseaux doivent être physiquement séparés entre eux par un routeur

Adresses particulières :

Le ou les octets désignant le réseau peuvent prendre des valeurs particulières :

– 0 : adresse des machines ignorant leur réseau
– 127 : adresse de loopback. L'interface de bouclage dont l'adresse IP est par convention **127.0.0.1** et le nom **localhost**, permet à un client et à un serveur situés sur la même machine de communiquer via TCP/IP.
– 255 : adresse de diffusion (broadcast)

Pour les octets qui désignent l'adresse locale, ce sont :

– 0 : adresse des machines ignorant leur adresse réseau (0.0.0.0) ; c'est le cas des stations sans disque au moment du boot.
– 255 : adresse de diffusion. (si l'adresse local correspond à plus ou moins d'un octet, l'adresse de diffusion correspond à l'adresse locale dont tous les bits ont été mis à 1).

Plan d'adressage : Un site qui souhaite mettre en place un réseau local dont l'usage est limité au site lui-même (réseau non relié à l'Internet), peut choisir des adresses IP des réseaux de classe A, B ou C de son choix. Si ce site désire ultérieurement se connecter à l'Internet, il devra :

– soit demander un réseau officiel et renuméroter toutes ses machines.
– soit demander une adresse officielle pour une de ses machines qui sera utilisée comme machine coupe-feu (firewall) et qui relaiera les communications des autres machines de son réseau.

1.2.1.5 Le fichier */etc/hosts* sous Unix

Ce fichier fait l'association entre les noms des machines et leur adresse IP, les noms étant plus faciles à manipuler que des adresses pour les utilisateurs. Chaque ligne du fichier contient :

– l'adresse Internet comme premier champs
– le nom de la machine comme deuxième champs
– des alias éventuels sur les champs suivants pour désigner la même machine par plusieurs noms différents.

Exemple de fichier hosts :

```
#
# Internet host table
#
127.0.0.1    localhost
#
195.1.1.1    mercure
195.1.1.2    venus serveur-www
195.1.1.3    mars  serveur-ftp
195.1.1.4    jupiter-eth0
#
196.10.10.1  saturne
196.10.10.2  jupiter-eth1
196.10.10.3  neptune-eth0
#
197.20.20.1  pluton
197.20.20.2  neptune-eth1
197.20.20.3  uranus
```

Ce fichier est à l'échelle d'un réseau local. Nous verrons que la conversion nom/adresse IP peut être effectuée par d'autres moyens (DNS, NIS).

1.2.1.6 La commande ifconfig

Afin de s'affranchir des nombreuses différences entre les équipements divers utilisés dans un environnement réseau, TCP/IP définit une interface abstraite, à travers laquelle on accède à la partie matérielle. Cette interface permet de gérer l'envoi et la réception de paquets.

Pour chaque périphérique à connecter au réseau, il faut une interface correspondante dans le noyau du système. Par exemple, les cartes ethernet sous Linux s'appellent *eth0* et *eth1*.

Afin d'être utilisable sur un réseau TCP/IP, une interface doit se voir attribuer une adresse IP, qui permet de l'identifier lors de communications avec le reste du monde. D'autres paramètres peuvent être ajustés. Voici la syntaxe de la commande et l'explication de quelques options possibles :

`ifconfig` *`interface`* `[`*`adresse`* `[`*`paramètres`*`]]`

L'argument *interface* est le nom de l'interface et adresse est l'adresse IP à lui assigner. Si ifconfig est invoquée uniquement avec le nom de l'interface, cette commande affiche alors la configuration courante de l'interface en question. Le nom de l'interface peut être remplacé par -a pour désigner l'ensemble des interfaces de la machine. Quelques paramètres parmi ceux connus par ifconfig :

- up : cette option marque l'interface comme étant accessible à la couche réseau du noyau. Elle est implicite lorsqu'une adresse est donnée sur la ligne de commande.
- down : Marque l'interface comme étant inaccessible à la couche IP du noyau.
- netmask *masque* : Assigne le masque de sous réseau à utiliser pour cette interface.
- broadcast *adresse* : Assigne l'adresse de diffusion qui est généralement constituée à partir de la valeur réseau en mettant tous les bits de la partie hôte à 1.

Elle est, en principe lancée au moment du démarrage de la machine pour configurer toutes ses interfaces (script /etc/rc.d/rc2.d/S10network sous Linux).
Exemple de ifconfig sur une machine Linux :

```
serveur-linux.lri.fr*/etc/rc.d/rc2.d >ifconfig
lo        Link encap :Local Loopback
          inet addr :127.0.0.1  Bcast :127.255.255.255  Mask :255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU :3584  Metric :
          RX packets :293053 errors :0 dropped :0 overruns :0 frame :0
          TX packets :293053 errors :0 dropped :0 overruns :0 carrier :0
          collisions :0

eth0      Link encap :Ethernet  HWaddr 00 :C0 :4F :8A :C5 :D3
          inet addr :129.175.7.100  Bcast :129.175.7.255  Mask :255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU :1500  Metric :1
          RX packets :5770455 errors :0 dropped :0 overruns :0 frame :0
          TX packets :6867983 errors :0 dropped :0 overruns :0 carrier :1
          collisions :1655116
          Interrupt :14 Base address :0xcc00
```

1.2.2 ARP

TCP/IP fonctionne avec des couches liaisons de données, comme Ethernet ou token-ring, qui ont leur propre schéma d'adressage. Au sein d'un réseau local Ethernet, lorsqu'une trame est envoyée d'une machine à une autre, c'est l'adresse Ethernet sur 48 bits qui détermine à quelle interface la trame et destinée. Le pilote réseau ne se préoccupe jamais de l'adresse IP de destination contenue dans le datagramme IP. La résolution d'adresse procure une correspondance entre deux formes différentes d'adresse : des adresses IP sur 32 bits et les adresses utilisées par la couche de liens de données.
Le protocole ARP (Address Resolution protocol) fournit une correspondance dynamique entre l'adresse IP et l'adresse matérielle correspondante. Le protocole RARP (Reverse Address Resolution Protocol) établit le processus inverse pour des systèmes dépourvus d'unités de disque.

1.2.2.1 Fonctionnement de ARP

Lorsqu'un utilisateur envoie la commande **ftp serveur-ftp**, cela déclenche les évènements suivants :

1. Le client FTP appelle la primitive *gethostbyname* pour convertir le nom de la machine en adresse IP en utilisant le DNS (Domain Name System) ou le fichier /etc/hosts.
2. Le client FTP demande à la couche TCP d'utiliser cette adresse IP.
3. TCP envoie une requête de connexion à la machine éloignée en émettant un datagramme à son adresse IP.
4. Si la machine se trouve sur un réseau local, le datagramme peut lui être envoyé directement. Sinon la fonction de routage détermine l'adresse IP d'un routeur de saut suivant.
5. Si la machine est sur un réseau Ethernet, elle doit convertir l'adresse IP sur 32 bits en une adresse Ethernet sur 48 bits. Cette opération est réalisée grâce à ARP.
6. ARP envoie une requête ARP sous forme d'une trame Ethernet à chaque machine du réseau (broadcast) dans laquelle se trouve l'adresse IP de la machine destinatrice.
7. La couche ARP de la machine de destination, après avoir vérifié que l'adresse IP contenue dans la requête était bien la sienne, envoie une réponse ARP contenant sa propre adresse Ethernet.
8. La réponse ARP est reçue par la machine qui avait émis la requête.
9. Le datagramme IP est envoyé à la machine destination.

Un cache ARP dans chacune des machines permet d'éviter des requêtes ARP pour des adresses fréquement utilisées. Il est important de préciser que le protocole ARP est implémenté directement au dessus de la couche liaison du réseau utilisé. Bien entendu il ne peut pas utiliser IP.

1.2.2.2 La commande arp

La commande arp permet de définir manuellement la correspondance entre adresse Internet et adresse Ethernet et de lister la table des correspondances connues par le système.
L'option -a permet de visualiser le contenu du cache ARP :

```
>arp -a
sun7j (129.175.7.80) at 08 :00 :20 :19 :22 :89 [ether] on eth0
cisco-7 (129.175.7.1) at 00 :10 :0B :A5 :20 :10 [ether] on eth0
sun-archi (129.175.7.2) at 08 :00 :20 :88 :9D :0B [ether] on eth0
pc-archi (129.175.7.20) at 00 :A0 :24 :7A :45 :95 [ether] on eth0
sun-parall (129.175.7.12) at 08 :00 :20 :9A :78 :AA [ether] on eth0
sun7g (129.175.7.77) at 08 :00 :20 :18 :67 :BF [ether] on eth0
```

L'option -s permet d'associer une adresse Ethernet à une adresse IP et permet donc d'atteindre une machine qui ne répond pas aux messages de diffusion ARP.

1.2.3 ICMP

ICMP (Internet Control Message protocol) est un protocole qui communique des messages qui réclament l'attention et notamment les messages d'erreur. Ces messages sont encapsulés dans des datagrammes IP et ont comme format celui décrit par la figure 1.8.
Les quatre premiers octets possèdent le même format quel que soit le message, le reste différant selon le message.
Le champ *type* indique le type du message ICMP et code le contexte dans lequel il est envoyé.
Le champ *somme de contrôle* (checksum) est calculé sur l'intégralité du message ICMP.

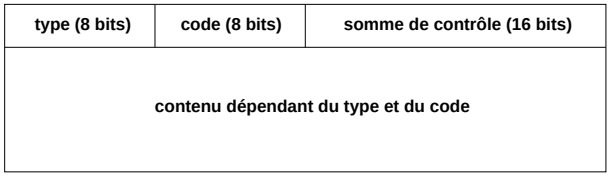

FIG. 1.8 – Message ICMP

1.2.3.1 Types de message ICMP

Le tableau 1.1 liste les différents types de message ICMP tels qu'ils sont définis par les champs type et code.

Un message d'erreur ICMP contient l'en-tête IP et les huit premiers octets du datagramme IP qui a provoqué l'erreur. Cela permet au module ICMP d'associer le message qu'il reçoit à un protocole particulier (TCP ou UDP indiqué dans le champ protocole de l'en-tête IP) et à un processus utilisateur particulier (à partir des numéros de port TCP ou UDP contenus dans les huit premiers octets du datagramme IP).

Certains évènements ne génèrent jamais de messages d'erreur ICMP. Ce sont :

- Un message d'erreur ICMP. Si tel n'était pas le cas, une erreur pourrait provoquer une erreur qui à son tour provoquerait une erreur et ainsi de suite. Par contre une erreur ICMP peut être générée en réponse à une requête ICMP.
- un datagramme destiné à une adresse IP de broadcast ou une adresse IP multicast.
- un datagramme envoyé en tant que broadcast de couche de liens.
- un fragment autre que le premier.
- un datagramme dont l'adresse source n'a pas été émise par une adresse unique (l'adresse source ne peut pas être une adresse 0, une adresse de bouclage, une adresse de broadcast ou une adresse multicast).

Ces règles permettent d'éviter les "orages de broadcast" .

1.2.3.2 La commande ping

La commande ping (dont le nom provient de "Packet Internet Groper") permet de vérifier l'accessibilité d'une machine. Le programme envoie un message de requête 'ICMP echo' en direction d'une machine et attend qu'elle lui retourne une réponse 'ICMP reply'. ping peut être utilisé comme moyen de diagnostic dans la mesure où vous ne serez en principe pas capable d'utiliser le moindre service d'une machine si vous ne pouvez la joindre avec ping. C'est donc une première approche pour déterminer la source d'un problème. ping est le client qui envoie des requêtes, le serveur étant la machine destinatrice. Le serveur ping n'est pas un processus utilisateur mais est directement implémenté dans le noyau.

Le serveur retourne en echo les champs identificateur et numéro de séquence, le premier contenant l'ID du processus émetteur et le second contenant un numéro qui commence à 0 et qui est incrémenté de 1 à chaque nouvelle requête echo envoyée. ping affiche ainsi le numéro de séquence de chaque paquet retourné permettant de voir si des paquets sont manquants.

ping envoie par défaut une requête par seconde et affiche en echo chaque réponse retournée. Cependant, certaines implémentations de ping nécessitent l'option -s pour ce mode de fonctionnement, se contentant d'envoyer par défaut une seule requête echo et d'afficher 'host is alive' si une réponse est reçue en echo, 'no answer' si aucune réponse ne parvient dans un délai de 20 secondes.

Exemple de ping :

type	code	description
0	0	réponse echo (cf ping)
3		destination inaccessible :
	0	réseau inaccessible
	1	machine inaccessible
	2	protocole inaccessible
	3	port inaccessible
	4	fragmentation nécessaire mais bit de fragmentation à 0
	5	échec de la route source
	6	réseau de destination inconnue
	⋮	
4	0	débit trop élevé
5		redirigé
8	0	requête echo (cf ping)
9	0	avertissement de routeur
10	0	sollicitation de routeur
11		temps dépassé :
	0	TTL vaut 0 pendant le transit
	1	TTL vaut 0 pendant le réassemblage
12		problème de paramètre :
	0	mauvais entête IP
	1	option requise manquante
13	0	requête timestamp
14	0	réponse timestamp
17	0	requête de masque d'adresse
18	0	réponse de masque d'adresse

TAB. 1.1 – Certains types de message ICMP

```
>ping ftp.inria.fr
PING ftp.inria.fr (192.93.2.54) : 56 data bytes
64 bytes from 192.93.2.54 : icmp_seq=0 ttl=249 time=4.2 ms
64 bytes from 192.93.2.54 : icmp_seq=1 ttl=249 time=4.2 ms
64 bytes from 192.93.2.54 : icmp_seq=2 ttl=249 time=6.3 ms
--- ftp.inria.fr ping statistics ---
12 packets transmitted, 12 packets received, 0% packet loss
round-trip min/avg/max = 4.0/4.3/6.3 ms
```

Lorsque la réponse à l'echo ICMP est renvoyée, le numéro de séquence s'affiche suivi par le TTL (Time-To-Live de l'en-tête IP) et le calcul du temps d'aller et retour.

1.2.4 Routage

Le routage est l'une des fonctions les plus importantes du protocole IP. La couche IP gère une table de routage en mémoire qu'elle consulte à chaque fois qu'elle reçoit un datagramme à envoyer. Lorsqu'un datagramme est reçu au moyen d'une interface réseau, IP vérifie si l'adresse de destination est l'une de ses propres adresses IP ou une adresse IP de broadcast. Dans ce cas, le datagramme est délivré au module de protocole indiqué par le champ protocole de l'en-tête IP. Dans le cas contraire, si la couche IP est configuré pour router, le paquet est réémis, sinon il est rejeté.
Chaque entrée dans la table de routage contient les informations suivantes :

- Adresse IP de destination (machine ou réseau)
- Adresse IP d'un routeur de saut suivant ou d'un réseau connecté directement.
- Des flags qui indiquent si l'adresse IP de destination est l'adresse d'un réseau ou d'une machine
- Le nom de l'interface réseau à laquelle le datagramme doit être passé pour être transmis.

Le routage IP est effectué sur la base du saut à saut. IP ne connaît la route complète d'aucune destination (sauf, bien sûr les destinations directement reliées à la machine émettrice).
IP interroge ses tables de routage dans l'ordre qui suit :

1. Recherche dans la table de routage de l'entrée correspondant à l'adresse intégrale du destinataire.
2. Recherche dans la table de routage de l'entrée correspondant exactement à l'identificateur de réseau du destinataire.
3. Recherche dans la table de routage d'une entrée dénommée *default*.

La complexité des tables de routage dépend de la topologie des réseaux auxquels la machine a accès :

1. Le cas le plus simple est celui d'une machine qui n'est pas connectée du tout au réseau. Les protocoles TCP/IP peuvent être utilisés pour une communicaton en locale. La table de routage se résume alors en une seule entrée pour l'interface de bouclage.
2. Le cas suivant est une machine d'un réseau local ne pouvant accéder qu'aux machines qui y sont connectées. La table de routage consiste en deux entrées : l'une pour l'interface de bouclage, l'autre pour le réseau local.
3. Un niveau supérieur est atteint lorsque d'autres réseaux deviennent accessibles au travers d'un seul routeur. La table de routage contient alors une entrée par défaut pointant vers ce routeur.
4. le niveau ultime est atteint lorsqu'on ajoute encore d'autres réseaux permettant ainsi l'existence de routes supplémentaires.

1.2.4.1 La commande netstat

Cette commande permet d'afficher les statistiques de fonctionnement des interfaces. Voici quelques une des options de netstat.

- -i affiche les statistiques des interfaces IP.
  Exemple :

```
>netstat -i
Kernel Interface table
Iface   MTU Met  RX-OK RX-ERR RX-DRP RX-OVR   TX-OK TX-ERR TX-DRP TX-OVR Flags
lo     3584   0 293127      0      0      0  293127      0      0      0 BLRU
eth0   1500   0 5774423     0      0      0 6870284      0      0      0 BRU
```

- -m affiche l'utilisation des tampons du réseau IP
- -n remplace les noms de machine par leur adresse IP
- sans options, netstat affiche les connexions en cours. Elles apparaissent comme un couple adresse IP/ numéro de port. Lorsque le port est défini dans /etc/services (cf 1.3.1), le service apparaît par son nom.
- -r permet d'afficher les tables routage.
  Exemple :

```
>netstat -r
Routing Table :
   Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
ups-lri_11           sun-asspro           U         3 441934  hme0
default              cisco-11             UG        0 973904  hme0
localhost            localhost            UH        0  13359  lo0
```

- Les lettres de la colonne *Flags* ont la signification suivante :
  - U indique que l'interface est up (la route est en service)
  - G indique que la machine située dans la colonne *Gateway* est le routeur permettant d'atteindre la machine ou le réseau qui se trouve dans la colonne *Destination*.
  - H associé à G indique que la route fait référence à une autre machine.
  - D indique que la route a été crée par un message "redirection ICMP".

Il existe différentes techniques de routage que nous allons passer en revue en prenant des exemples sur la configuration du réseau de la figure 1.9 :

1.2.4.2 Le routage statique

L'administrateur défini à la main les tables de routage et les saisit une fois pour toute. Cette méthode est plutôt destiné pour des réseaux locaux de petite taille et relativement stables.

1.2.4.3 Le routage directe

Pour chaque machine, la route directe est automatiquement créée par la commande ifconfig utilisée pour configurer l'interface. Exemple de déclaration pour la machine jupiter :

```
>ifconfig eth0 195.1.1.4   netmask 255.255.255.0
>ifconfig eth1 196.10.10.2 netmask 255.255.255.0
```

La machine jupiter peut afficher les routes directes vers les deux réseaux locaux auquels elle est connectée avec la commande netstat :

```
>netstat -nr
Kernel IP routing table
Destination   Gateway          Genmask         Flags   MSS Window  irtt Iface
195.1.1.0     195.1.1.4        255.255.255.0   U      1500 0          0 eth0
196.10.10.0   196.10.10.2      255.255.255.0   U      1500 0          0 eth1
127.0.0.0     0.0.0.0          255.0.0.0       U      3584 0          0 lo
```

Ces routes indiquent que pour atteindre les machines du réseau 195.1.1, jupiter doit utiliser l'interface eth0 d'adresse IP 195.1.1.4 et que pour atteindre les machines du réseau 196.10.10, jupiter doit utiliser l'interface eth1 d'adresse IP 196.10.10.2. Les routes directes correspondent donc aux réseaux auquels une machine est directement connectée.

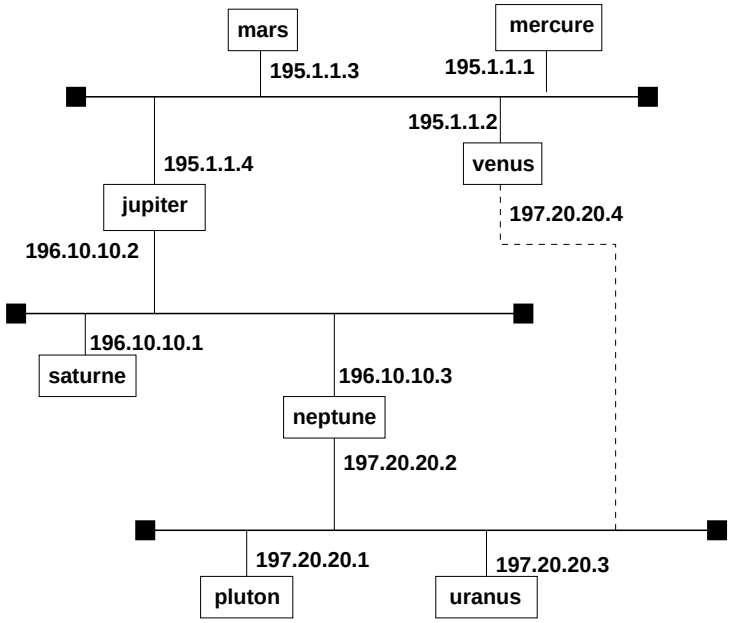

FIG. 1.9 – Exemple de réseau local

1.2.4.4 Le routage indirect

Il s'agit pour une machine d'utiliser une autre machine comme routeur pour atteindre sa destination. On le réalise en ajoutant une nouvelle route à l'aide de la commande route. Par exemple on va ajouter une route pour que mercure puisse atteindre saturne en passant par jupiter :

```
route add saturne jupiter-eth0 1
```

ou

```
route add host 196.10.10.1 195.1.1.4 1
```

Pour compléter l'opération, il faut que saturne puisse répondre à mercure et donc qu'elle exécute la commande :

```
route add mercure jupiter-eth1 1
```

ou

```
route add host 195.1.1.1 196.10.10.2 1
```

On peut voir que la commande route peut utiliser les noms ou les adresses IP. Le dernier argument de la commande route donne le nombre de sauts que doivent effectuer les paquets IP pour parvenir à leur destination (c'est-a-dire le nombre de routeurs à traverser). Toute autre valeur que 0 indique donc un routage indirect. A noter que jupiter ayant 2 interfaces et deux adresses IP, elle a aussi deux noms.
On peut vérifier la route créée sur saturne avec netstat :

```
>netstat -nr
Destination           Gateway             Flags  Ref   Use   Interface
--------------------- ------------------- ----- ----- ------ ---------
195.1.1.1             196.10.10.2         UGH       3 441934  hme0
```

On peut voir les flags G pour routeur et H pour routage machine.
Les commandes *route* doivent être rajoutées dans les scripts de démarrage pour que le routage soit correct après le "boot" de la machine. Ce type de routage est un routage machine car il précise que pour atteindre la machine A, il faut envoyer les trames à la machine B. Il peut être plus judicieux de dire : pour atteindre le réseau sur lequel se trouve la machine A, envoyer les trames à la machine B. Cela se traduit par la commande :

```
route add net 196.10.10 195.1.1.4 1
```

Cette forme de routage, appelée routage réseau, permet de diminuer le nombre de commandes à effectuer (puisque dans le premier cas, il faut autant de commandes route qu'il y a de machines sur le réseau 196.10.10) et prend en compte la mise en place de futures machines.
La premiere forme peut néanmoins présenter un intérêt en terme de sécurité si l'on veut limiter le nombre de machines du réseau 196.10.10 joignables à partir du reseau 195.1.1.
Il faut noter qu'un dialogue ne peut s'établir entre deux machines que si chacune des deux machines déclare le routage permettant d'atteindre l'autre.
Quand on regarde ce qui se passe au niveau des trames, on remarque que lorsque mercure veut communiquer avec saturne, il envoie des trames qui ont comme adresse ethernet destination, l'adresse ethernet du routeur jupiter (déterminée grace à la table de routage de mercure). Le module ethernet de jupiter va remonter la trame au niveau IP et s'apercevoir que le paquet ne lui est pas destiné, l'adresse IP de destination étant celle de saturne. jupiter va donc, grâce à sa table de routage, renvoyer la trame avec comme adresse ethernet de destination celle de saturne. Une réponse de saturne à mercure se traduit par un acheminement inverse du paquet IP.
Prenons maintenant le cas où la machine mercure veut dialoguer avec la machine pluton :
mercure doit déclarer (entre autres)

```
route add host 197.20.20.1 195.1.1.4 2
```

et pluton

```
route add host 195.1.1.1 197.20.20.2 2
```

Les routeurs doivent aussi déclarer les routes suivantes. Pour jupiter :

```
route add net 197.20.20 196.10.10.3 1
```

Pour neptune :

```
route add net 195.1.1 196.10.10.2 1
```

On remarque que le routage statique demande une gestion assez rigoureuse des tables de routage et convient donc surtout aux réseaux stables.

1.2.4.5 Le routage par redirection ICMP

Il s'agit pour une machine de déclarer un routeur par défaut qui centralisera alors la décision de routage. Lorsque la machine ne peut trouver une route, elle envoie le paquet au routeur par défaut. Dans notre exemple, si nous prenons jupiter comme routeur par défaut, mercure peut déclarer :

```
route add defaut jupiter-eth0 1
```

et se fier à jupiter pour l'acheminement des paquets vers leur destination. Imaginons maintenant que la machine venus soit aussi connectée au réseau 197.20.20 (comme indiqué sur la figure 1.9 avec les pointillés). La route par défaut en passant par jupiter n'est alors plus la bonne solution pour atteindre pluton ou uranus.

Le principe de la redirection est le suivant :

1. mercure envoie un datagramme IP à jupiter pour atteindre uranus, ce qui est cohérent puisque jupiter est le routeur par défaut.

2. jupiter reçoit le datagramme, interroge sa table de routage et détermine que venus est le routeur de saut suivant adéquat à qui il doit envoyer le datagramme. Au moment d'émettre le datagramme vers venus, il s'aperçoit qu'il le fait avec la même interface qui lui avait permis de le recevoir. Cela constitue alors la preuve pour le routeur qu'il est capable d'émettre une redirection vers l'émetteur d'origine.
3. jupiter émet une redirection ICMP à mercure lui indiquant de passer par venus plutôt que par jupiter pour atteindre sa destination finale.

On peut vérifier la création de la route par le module IP avec netstat :

```
>netstat -nr
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
197.20.20.3          195.1.1.2            UDGH     3 441934  hme0
...
```

Le flag D indique que la route a été créée par un message "ICMP redirect".

1.2.4.6 Le routage dynamique

Le routage statique que l'on vient de voir s'applique pour des réseaux de taille modeste, qui ne comportent qu'un seul point de connexion vers les autres réseaux et qui n'incluent aucune route redondante. Dans les autres cas, le routage dynamique s'avère le plus adapté. Le routage dynamique met en oeuvre un protocole de communication inter-routeurs, chacun d'eux informant ses voisins des réseaux auxquels il est connecté. Les démons de routage (qui exécutent le protocole de routage sur le routeur) mettent à jour la table de routage du noyau avec l'information qu'ils recoivent des routeurs adjacents.

Le mécanisme de routage que nous avons décrit dans les sections précédentes est toujours valable avec le routage dynamique. Par contre, le démon de routage donne au système le moyen de mettre en oeuvre une politique de routage lui permettant de choisir quelles routes placer dans la table de routage du noyau. Si le démon se trouve confronté à plusieurs routes conduisant à la même destination, il choisit celle qui lui semble la meilleure et l'insère dans la table de routage. Si le démon s'aperçoit qu'une liaison a été rompue, il est capable d'effacer les routes affectées et de choisir des routes alternatives.

De nombreux protocoles de routage différents sont actuellement utilisés au sein d'Internet. L'Internet est organisé sous la forme d'un ensemble de systèmes autonomes (ou AS pour Autonomous systems), chacun d'eux étant administré par une seule entité. Chaque système autonome est en mesure de choisir son propre protocole de routage permettant le dialogue inter-routeurs au sein du système. Ce protocole est appelé *Interior Gateway Protocol* (IGP). Le plus répandu des IGP est le protocole RIP (Routing Information Protocol). OSPF (Open Shortest Path First Protocol) est un IGP apparu récemment et destiné à remplacer RIP. Des mauvaises langues disent que RIP veut dire "rest in peace".
Des protocoles de routage distincts, appelés *Exterior Gateway Protocol* (EGP) sont utilisés entre les routeurs appartenant à des systèmes autonomes différents. L'EGP prédominant a été un protocole qui s'appelle aussi EGP. Un nouvel EGP appelé *Border Gateway Protocol* est destiné à remplacer EGP.

1.2.4.7 L'algorithme de routage par vecteur de distances

RIP implémente l'algorithme de routage par vecteur de distance que nous présentons maintenant. Le réseau est modélisé par un graphe, dont les sommets sont les machines dans le réseau et il y a une arête entre deux sommets si les machines peuvent communiquer directement sans passer par un routeur. Le poids de l'arête correspond au coût de la liaison, et va être simplement 1 dans RIP. C'est à dire que RIP calcule les tables de routage en minisant le nombre de réseaux — et donc aussi le nombre de routeurs — à traverser. Le poids d'un chemin est naturellement la somme du poids des arêtes. Vous conaissez l'algorithme de plus court chemin de Dijkstra. Mais cet algorithme diffère dans le sens où il est

exécuté de manière distribué sur tous les sommets, et aucun sommet n'a à connaitre le graphe complet. Un sommet ne connait que ses voisins immédiats et le poids des arrêtes qui les relient. Pour simplifier la présentation, on supposera qu'un sommet connait aussi le nom de tous les autres sommets.

Chaque sommet dispose de deux vecteurs *dist* et *succ* dont les indices sont des noms de sommets. Ils associent à chaque sommet destinataire la prochaine passerelle par laquelle il faut router les paquets pour cette destination et une distance, que le sommet croit être la longueur du plus court chemin vers cette destination. A chaque coup d'horloge — toutes les 30 secondes dans RIP — les sommets envoient à leur voisins les vecteurs de distances. Lorsqu'un sommet reçoit un vecteur de distances de son voisin il le considère comme une offre pour router par lui. Pour chaque destination il va alors comparer les offres qu'il a obtenues des voisins et choisir le moins coûteux. Cette version, qu'on appelle *sans clivage d'horizon*, a un problème. Lorsqu'un sommet devient inaccessible, la mauvaise nouvelle se propage très lentement, et pendant un moment les tables de routage ne cessent de changer. En même temps les distances affichées vers ce sommet augmentent lentement (voir l'exemple donné dans les transparents du cours). On définit ∞ comme étant toute valeur dépassant la plus grande distance possible dans le graphe, ce qui garantit la stabilité de l'algorithme. Dans RIP toute valeur supérieure ou égale à 16 est considérée comme infinie. RIP n'est donc pas utilisable sur de très grands réseaux.

Pour accélerer la stabilisation de l'algorithme, il existe une petite modification de l'algorithme, qu'on appelle *algorithme de vecteur de distance avec clivage d'horizon*, qui consiste tout simplement à ne pas faire d'offre à un voisin pour les destinations pour lesquelles c'est justement ce voisin qui est la passerelle. Dans cette version, les mauvaises nouvelles se propagent de nouveau convenablement sur des graphes qui sont sans cycles. Par contre pour des graphes avec cycle, la propagation des mauvaises nouvelles reste lente.

1.2.4.8 Le programme traceroute

La commande traceroute permet de déterminer les routeurs franchis pour accéder à la machine donnée en argument. Exemple :

```
#traceroute ftp.inria.fr
traceroute to ftp.inria.fr (192.93.2.54), 30 hops max, 40 byte packets
 1  cisco-7 (129.175.7.1)  0.809 ms  0.676 ms  0.639 ms
 2  ups-routeur (193.55.18.193)  1.143 ms  0.967 ms  0.893 ms
 3  195.83.240.209 (195.83.240.209)  1.156 ms  1.102 ms  1.009 ms
 4  stlambert1.rerif.ft.net (193.48.53.221)  2.477 ms  1.929 ms  1.975 ms
 5  inria-rocquencourt-atm.rerif.ft.net (193.48.53.226)  3.901 ms  3.753 ms  3.752 ms
 6  rocq-gw.inria.fr (192.93.122.2)  4.136 ms  4.164 ms  3.942 ms
 7  ftp.inria.fr (192.93.2.54)  4.313 ms  4.000 ms  3.869 ms
```

La commande traceroute opère en envoyant un paquet IP avec un champs *Time To Live* (TTL) à 1. Le premier routeur envoie donc un message ICMP indiquant que le message ne peut pas être retransmis plus loin puisque son TTL est épuisé. On détermine ainsi l'adresse du premier routeur. Le paquet est ensuite renvoyé avec un TTL à 2 et ainsi de suite en incrémentant la champs TTL pour déterminer la liste des routeurs traversés pour parvenir à la machine indiquée.

1.2.5 IPV6

Actuellement la taille de l'Internet double tous les ans. Il y a deux problèmes à résoudre :
- L'épuisement des adresses IP
- L'explosion des tables de routage

Un nouveau protocole - IP version 6 (IPV6) - donne les améliorations suivantes :
- adressage d'un espace beaucoup plus grand
- un routage plus efficace

Ce nouveau protocole dont l'implémentation de la majeure partie est réalisée a, entre autres, comme caractéristiques :

- une adresse de 16 octets ($340 * 10^36$ adresses potentielles) de type hiérarchique.
- Un en-tête simplifié pour une meilleure efficacité de la commutation des routeurs.
- Une extension de l'en-tête pour les options (qui ne sont plus limitées à 40 octets).

1.3 Couche transport : UDP/TCP

1.3.1 La notion de port

Les protocoles UDP et TCP s'appuient sur la notion de port de protocole. Au lieu de considérer que la destination finale d'une donnée est un processus de la machine destination, nous dirons qu'une donnée est envoyée vers une destination abstraite nommée le port. Le port fait abstraction du processus physique destinataire dans la mesure où les services destinataires doivent pouvoir être identifiés par leur fonction et non par les processus qui les réalisent.
Le port est caractérisé par un nombre entier positif attribué par le système d'exploitation à la demande du processus. Le système fournit un mécanisme d'interface que les processus utilisent pour avoir accès à un port.
Le fichier */etc/services* sur Unix donne la liste d'association entre numéros de port et services. Ce fichier dont un extrait est représenté par la table 1.2 contient la déclaration des services connus par les différentes commandes utilisant la famille de protocoles IP.
Lorsque l'on développe un nouveau service, il faut le rajouter dans ce fichier en lui attribuant un numéro de port qui n'est pas encore affecté. Les ports de 1 à 1023 sont des ports réservés. Un processus peut utiliser un port réservé seulement si son EUID *(user id)* est égal à 0, l'identificateur de *root*. Les ports inférieurs à 512 sont des ports attribués par les autorités Internet. On les appelle "ports bien connus" (well known ports). Ces numéros de ports sont en général utilisés par les démons pour écouter les demandes de connexion de leurs clients.

1.3.2 UDP

Le protocole UDP (User Datagram Protocol) permet aux applications d'échanger des datagrammes. Le protocole UDP utilise la notion de "port" qui permet de distinguer les différentes applications qui s'exécutent sur une machine. En plus du datagramme et de ses données, un message UDP contient, à la fois, un numéro de port source et un numéro de port destination.

UDP fournit un service en mode non connecté et sans reprise sur erreur. Il n'utilise aucun acquittement, ne reséquence pas les messages et ne met en place aucun contrôle de flux.

Il correspond au niveau transport de l'architecture de référence mais c'est un protocole particulièrement simple. Son avantage est un temps d'exécution court qui permet de tenir compte des contraintes de temps réel ou de limitation de place sur un processeur.
Beaucoup d'applications qui n'ont pas besoin d'une sécurité très forte au niveau transmission ainsi que les logiciels de gestion qui demandent des interrogations rapides de ressources, utilisent UDP.

Les datagrammes UDP sont, bien sur, encapsulés dans des datagrammes IP.

1.3.2.1 En-tête UDP

La figure 1.10 présente les champs constituant l'en-tête UDP.

- Les *numéros de port* identifient les processus émetteur et récepteur.
- Le champs *longueur UDP* contient la longueur de l'en-tête et la taille des données UDP en octets (au minimum 8). Cette longueur est redondante car IP contient sa longueur totale en octets.
- Le champs *Somme de contrôle* est un calcul sur l'en-tête et les données (contrairement à IP qui n'intègre que l'en-tête).

```
#ident "@(#)services   1.16    97/05/12 SMI"   /* SVr4.0 1.8   */
#
# Network services, Internet style
#
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver

...

#
# UNIX specific services
#
# these are NOT officially assigned
#
exec            512/tcp
login           513/tcp
shell           514/tcp         cmd             # no passwords used
printer         515/tcp         spooler         # line printer spooler
courier         530/tcp         rpc             # experimental
uucp            540/tcp         uucpd           # uucp daemon
biff            512/udp         comsat
who             513/udp         whod
syslog          514/udp
talk            517/udp
route           520/udp         router routed
```

TAB. 1.2 – Le fichier /etc/services

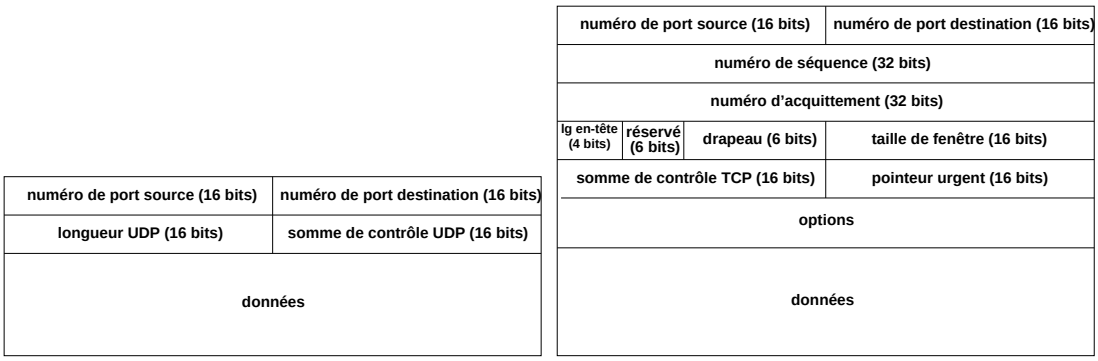

FIG. 1.10 – En-têtes UDP et TCP

1.3.3 TCP

TCP (Transmission Control Protocol) est un service de transport fiable. Pour arriver à cette fonctionnalité, TCP définit un certain nombre de caractéristiques :

- flot d'octets : les données échangées sont vues comme un flot de bits, scindé en octets reçus dans l'ordre où ils ont été envoyés.
- circuit virtuel en mode connecté : le transfert des données ne peut commencer qu'après l'établissement d'une connexion entre les deux machines. Durant le transfert, les deux machines continuent à vérifier que les données sont transmises correctement. Le terme de circuit virtuel vient du fait que les deux programmes d'application voient la connexion comme un circuit physique, la fiabilité de la transmission étant une illusion créée par le service transport.
- transfert bufferisé : les programmes d'application envoient leurs données sur le circuit virtuel en les passant régulièrement au système d'exploitation de la machine. Chaque application choisit la taille des données qui lui convient. TCP est libre de découper les données en paquets de tailles différentes de ce qu'il a reçu de l'application. Pour rendre le transfert plus performant, TCP attend d'avoir suffisamment de données pour remplir un datagramme avant de l'envoyer sur le réseau.
- flot de données non structurées : le service transport ne prend pas en compte les données structurées (c'est du ressort de l'application).
- connexion duplex : la connexion permet un transfert de données bidirectionnel. Ce sont deux flots de données inverses, sans interaction apparente. Ce principe permet de renvoyer des acquittements d'un sens de transmission alors que les données arrivent de l'autre sens.

Le protocole TCP définit la structure des données et des acquittements échangés, et les mécanismes permettant de rendre le transport fiable. Il spécifie comment distinguer plusieurs connexions sur une même machine, et comment détecter et corriger une perte ou une duplication de paquets. Il définit comment établir une connexion et comment la terminer.

1.3.3.1 En-tête TCP

Les données TCP sont encapsulées dans des datagrammes IP. La figure 1.10 montre le format de l'en-tête. Sa taille est de 20 octets sauf présence d'options.

- Chaque segment TCP contient le *numéro de port source* et le *numéro de port destination* afin d'identifier l'application émettrice et receptrice. Ces deux valeurs combinées aux adresses IP source et destination contenues dans l'en-tête IP, identifient chaque connexion de façon unique. La combinaison d'une adresse IP et d'un numéro de port est appelé une socket. C'est également le nom de l'interface de programmation dérivée de Berkeley. C'est la paire de sockets (@IP/no de port client

et @IP/no de port serveur) spécifiant les deux points d'extrémité qui identifient chaque connexion TCP sur un internet.
- le *numéro de séquence* est utilisé pour rétablir l'ordre des segments reçus. Ce numéro est incrémenté par le nombre d'octets contenus dans le segment. Ainsi si chaque segment contient 500 octets de données, ce numéro prendra dans les segments successifs 0, 500, 1500, etc. Le numéro de séquence est donc un pointeur sur le flot de données.
- Le *numéro d'acquittement*, valide seulement si le bit ACK est à 1, acquitte un segment en désignant le numéro de séquence du prochain segment qui sera transmis par l'autre extrémité de la connexion. Un segment contenant un numéro d'acquittement de 1000 indique que les données jusqu'à l'octet 1000 ont été reçues. En principe, lorsque l'émetteur ne reçoit pas d'acquittement dans un délai raisonnable, il renvoie le segment. En réalité, pour des raisons d'efficacité, on n'attend pas que chaque segment soit acquitté pour envoyer le suivant. Le champ fenêtre est utilisé par chaque extrémité en indiquant combien de données elle est prête à recevoir.
- Le champ *longueur d'en-tête* donne la longueur de l'en-tête en mots de 32 bits. Ce champ est nécessaire car la longueur du champ options est variable.
- Le champ suivant comprend 6 bits de flag dont 1 ou plusieurs peuvent être mis à 1 au même instant. Ces flags sont les suivants :
  - URG : le segment est à traiter en priorité et le champ *pointeur urgent* est significatif.
  - ACK : le segment transporte un acquittement et le champs *numéro d'acquittement* est significatif.
  - PSH : le segment doit être transmis immédiatement à la couche supérieure sans attendre d'autres segments. Ce bit est utilisé pour les applicatifs intéractifs comme telnet ou rlogin de manière à recevoir l'écho des caractères dans un délai raisonnable.
  - RST : provoque la coupure immédiate de la connexion ou refuse une demande de connexion.
  - SYN : le segment est un segment permettant d'initialiser une connexion. Une demande de connexion positionne SYN à 1 et ACK à 0, son acquittement positionne SYN à 1 et ACK à 1.
  - FIN : le segment indique qu'il n'y a plus de données à transmettre et il termine donc la connexion.
- Le champ *taille de fenêtre* indique le nombre d'octets qui seront acceptés par l'émetteur du paquet à compter de l'octet désigné par le champ *numéro d'acquittement*.
- La *somme de contrôle TCP* recouvre l'en-tête et les données TCP. La méthode de calcul est identique à celle utilisée pour le protocole UDP.
- Le champ *pointeur urgent*, valide seulement si le bit URG est à 1, pointe sur les données qui doivent être traitées de façon prioritaire. Les données du début jusqu'à la donnée pointée sont concernées et doivent être délivrées au plus tôt à la couche de niveau immédiatement supérieure.
- Le champ *options* est facultatif et permet par exemple d'indiquer à la station destinataire, la longueur maximum des segments (MSS) à recevoir.

1.3.3.2 Etablissement et terminaison d'une connexion TCP

TCP étant un protocole orienté connexion, avant que l'une des extrémité puisse envoyer des données à l'autre, une connexion doit être établie entre elles :

1. L'extrémité émettant la requête (appelé le client) émet un segment SYN indiquant le numéro du port du serveur avec lequel le client veut se connecter et le numéro de séquence initial (NSI) du client. C'est le segment 1.
2. Le serveur répond avec son propre segment SYN contenant le numéro de séquence initial du serveur (segment 2). Le serveur acquitte aussi le SYN du client en acquittant le NSI du client plus un. Un SYN consomme un numéro de séquence.
3. Le client doit acquitter ce SYN à partir du serveur en acquittant le NSI du serveur plus un (segment 3).

Ces trois segments complètent l'établissement de la connexion et caractérisent la poignée de mains à trois voies (three-way handshake). Il faut quatre segments pour terminer une connexion. Puisqu'une

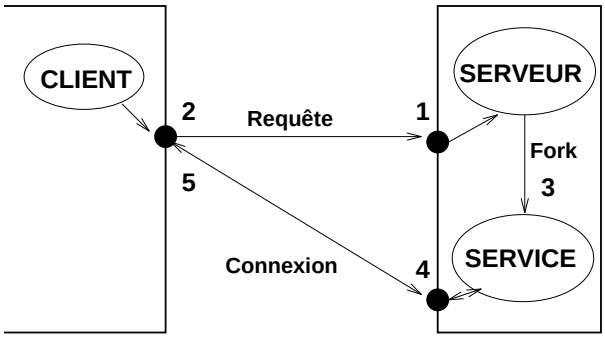

FIG. 1.11 – Connexion en mode client-serveur

connexion TCP est "full duplex" (les données peuvent s'écouler dans une direction indépendamment de l'autre direction), chaque direction peut être arrêtée indépendamment. Lorsque TCP reçoit un FIN, il doit notifier à l'application que l'autre extrémité a terminé son flux de données. L'émission d'un FIN est normalement l'aboutissement d'une application effectuant une fermeture. Un TCP peut encore envoyer des données après avoir reçu un FIN mais en pratique, peu d'applications tirent parti de cette semi-fermeture. L'extrémité qui effectue en premier la fermeture effectue la fermeture active alors que l'autre extrémité effectue la fermeture passive. Par exemple, quand un utilisateur tape *quit* lors d'une session telnet, le client TCP envoie un FIN fermant le flux de données du client vers le serveur. Lorsque le serveur reçoit le FIN, il envoie un ACK du numéro de séquence plus un.

1.3.3.3 Conception d'un serveur TCP

La plupart des serveurs TCP sont concurrents : lorsqu'une nouvelle requête de connexion arrive à un serveur, le serveur accepte la connexion et invoque un nouveau processus pour gérer le nouveau client. Sous Unix, la technique est de créer un nouveau processus en utilisant la primitive fork(). La figure 1.11 illustre ce procédé en mode client-serveur.

Nous allons voir avec la commande netstat comment TCP gère les numéros de port en prenant comme exemple des connexions sur un serveur telnet :

```
>netstat -a -n --inet
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp       0      0 0.0.0.0 :23             0.0.0.0 :*              LISTEN
```

La sortie a été obtenue sur un système ne comportant aucune connexion telnet active. Seules les lignes concernant le serveur telnet ont été conservées. Nous avons inscrit les options -a (affiche tous les points d'extrémité du réseau), -n (remplace les noms par les adresses IP) et –inet (affiche seulement les points d'extrémité UDP et TCP).
Le port local affiché est 23 et correspond au port réservé pour telnet. Nou voyons que le point d'extrémité est dans l'état LISTEN attendant qu'un connexion arrive.
Regardons maintenant le résultat de netstat après que nous ayons effectué une connexion telnet depuis la machine 129.175.11.100 :

```
>netstat -a -n --inet
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 129.175.7.100 :23        129.175.11.100 :54808    ESTABLISHED
tcp        0      0 0.0.0.0 :23              0.0.0.0 :*               LISTEN
```

La première ligne pour le port 23 est la connexion ESTABLISHED. Les quatre éléments pour la connexion sont remplis :

1. l'adresse IP locale : 129.175.7.100
2. le numéro de port local : 23
3. l'adresse IP distante : 129.175.11.100
4. le numéro de port distant : 54808

Le point d'extrémité dans l'état LISTEN est celui que le serveur utilise pour accepter les futures requêtes de connexion.

Chapitre 2

Les applications

2.1 Programmation socket en Java

Les sockets (développés par BSD) réalisent des communications inter-processus (IPC : Inter Process Communications). La démarche est la suivante : Le processus serveur ouvre un port en mode serveur qui va écouter un port donné. Puis il attendra que des clients veuillent bien s'y connecter. Une fois que la connexion est établie, les processus serveur et client peuvent communiquer via des flux, comme s'ils écrivaient à la sortie standard et lisaient de l'entrée standard. Le processus client ouvre un socket en mode client qui va se connecter sur une machine à un port donné.

```
import java.net.*;
import java.io.*;
public class Serveur {
  static public void main(String args[])
      throws IOException {
      ServerSocket s = new ServerSocket(8080);
      System.out.println("Ecoute du port 8080");
      Socket client = s.accept();
      System.out.println("Chouette un client se connecte");
      PrintWriter out = new PrintWriter(client.getOutputStream(),true);
      out.println("Désolé, mais je suis occupé");
      out.close();
      client.close();
  }
}
```

Dans cet exemple, un serveur écoute un port, en créant un ServerSocket avec le numéro de port à écouter. Puis la méthode accept() bloque jusqu'à ce qu'un client se connecte, puis retourne un Socket vers ce client. Pour écrire ou lire il faut ensuite créer des flux. Le plus simple est de créer un PrintWriter, ce qui permet d'écrire avec println exactement comme pour System.out. La création d'un flux pour lire est un peu plus compliquée, comme le montre l'exemple suivant, un client qui demande l'heure à un serveur.

```
import java.net.*;
import java.io.*;
public class Client {
  static public void main(String args[])
      throws IOException {
      System.out.print("Connexion au serveur ");
      Socket s = new Socket(args[0], 8080);
```

```
            System.out.println("réussie") ;
            BufferedReader in = new BufferedReader(
                new InputStreamReader(s.getInputStream())) ;
            String lheure = in.readLine() ;
            System.out.println("Il est exactement "+lheure) ;
            in.close() ;
            s.close() ;
        }
    }
```

Notez la gestion paresseuse des exceptions pour cet exemple. Aussi c'est une bonne chose de fermer d'abord les flux avant de fermer un socket, car ceci va vider les tampons à temps.

2.2 Le système de fichiers distribué NFS

NFS (Network File System), le système de fichiers par réseau est un service qui permet d'accéder aux fichiers présents sur des machines distantes exactement comme s'ils étaient locaux.

Ce service a été développé par SUN et a été repris par tous les Unix. NFS utilise le protocole UDP et les appels de procédures distantes Sun RPC (Remote Procedure Call).

Le serveur NFS va rendre accessible un système de fichiers à d'autres machines. Le client NFS essaie de monter un répertoire d'une machine distante sur un répertoire local, exactement de la même manière qu'il le fait pour un périphérique physique. Il utilise pour cela la commande mount qui permet de se connecter au démon mountd de la machine distante qui testera si la machine cliente est autorisée à faire cette requête. Lorsqu'un utilisateur accède ensuite à un fichier par NFS, le noyau envoie un appel RPC à nfsd (le démon NFS) sur le serveur.

2.2.1 Démons NFS

Des démons sont lancés côté serveur et côté client.

- **nfsd** : il tourne sur les serveurs NFS. Il répond aux requêtes NFS des clients.
- **mountd** : il tourne sur les serveurs NFS. Il traite les demandes de montage des processus clients.
- **lockd** : il tourne sur les serveurs et les clients. Ce démon gère les requêtes de verrou de fichiers.
- **statd** : il tourne sur les serveurs et les clients. C'est un démon d'observation réseau utilisé par lockd.

2.2.2 Partage des systèmes de fichiers

Pour chaque client, le serveur détermine le type d'accès autorisé. Cet accès est spécifié dans un fichier qui liste les répertoires partagés. Par défaut, mountd interdira à tout le monde de monter des répertoires de l'hôte local. Pour autoriser un ou plusieurs hôtes à monter un répertoire par NFS, ce répertoire doit être exporté. Les fichiers utilisés sont différents suivant les systèmes.

2.2.2.1 */etc/exports* et exportfs

Exemple de fichier exports sur Linux :

```
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
/usr            *.local.domain(ro) @trusted(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
/pub            (ro,insecure,all_squash)
/pub/private    (noaccess)
```

Entre parenthèses vous trouvez les options d'exportation. A titre d'exemple, *rw* permet un accès en lecture-écriture, *root_squash* va transformer tous les identificateurs UID/GID 0 vers le UID/GID anonyme. Ceci évite qu'un utilisateur root qui aurait monté ce répertoire puisse modifier des fichiers sensibles. Cette derniere option est mise par défaut.

2.2.3 Monter un volume NFS

Les volumes NFS sont montés presque de la même facon que les systèmes de fichiers traditionnels. La syntaxe est la suivante :

```
mount −t nfs machine_distante :répertoire_distant répertoire_local options
```

La notation utilisée étant unique, on peut omettre l'option -t nfs. Les options additionnelles peuvent être données sur la ligne de commande après le commutateur -o.
Vous pouvez ajouter des entrées dans le fichier */etc/fstab* pour indiquer des volumes qui seront montés soit au boot, soit sur demande.

Exemple d'une entrée dans fstab :

```
#volumepoint de montagetypeoptions
pc15 :/mail/var/spool/mailnfsintr
```

Ce volume peut ensuite être monté par la commande **mount /var/spool/mail**.

2.2.4 L'automonteur

L'"automounter" permet de monter dynamiquement des ressources NFS, c'est-à-dire qu'une partition distante sera montée sur une machine locale au moment où un utilisateur de cette machine essaiera d'y accéder. De la même manière, le démontage de la partition est automatique et a lieu après un temps paramétrable d'inutilisation.

C'est le démon **automount**, lancé au démarrage de la machine et configuré à l'aide du fichier */etc/auto_master*, qui va gérer ces montages dynamiques.

2.2.4.1 Le fichier /etc/auto_master

Ce fichier contient des lignes de la forme *point-de-montage map [options]* où
- *point-de-montage* indique le point de montage de la partition NFS.
- *map* indique le nom du fichier (ou map dans le cadre de NIS - cf 2.4 -) dans lequel sont décrits l'ensemble des ressources accessibles à travers le précédent point de montage.
- *options* indique les options de montage utilisées par défaut.

Exemple de fichier /etc/auto_master :

```
/special auto.special  -rw,intr,soft
/users auto.users    -rw,intr,soft,nosuid
/local auto.local    -rw,intr,soft
```

Un extrait de la map auto.special peut être, par exemple :

```
...
maple hostA :/softs/mapleVR4 hostB :/usr/local/mapleVR4
...
```

Un utilisateur du réseau, en accédant à /special/maple, provoquera le montage automatique de la ressource correspondante de hostA ou hostB (la première disponible).

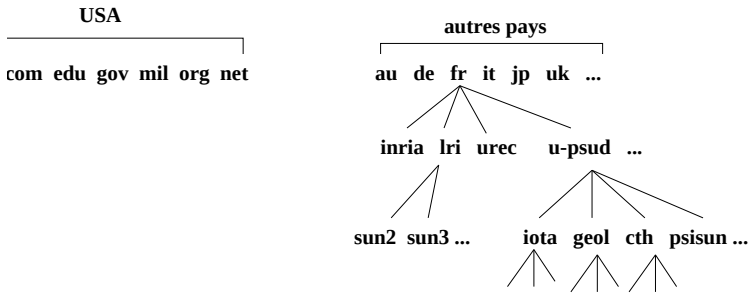

FIG. 2.1 – Organisation hiérarchique du DNS

2.3 DNS

Les adresses IP des machines d'un réseau local sont définies dans le fichier hosts. Ne pouvant y inclure la liste des machines du monde entier, la solution est de définir un nommage hiérarchique avec un système de serveur de noms (DNS : Domain Name System). Les informations sont automatiquement disséminées et uniquement auprès de ceux qui veulent les obtenir. Si un serveur DNS reçoit une requête portant sur des informations qu'il ne possède pas, il transmet la requête à un serveur faisant autorité, c'est-à-dire un serveur DNS chargé de maintenir les informations concernant le domaine demandé. Le serveur local conserve alors la réponse de ce serveur dans un cache pour répondre directement aux futures requêtes.
L'espace des noms est formé de domaines qui sont organisés en arborescence. Le DNS possède un domaine racine situé en haut de la hiérarchie du domaine servi par un groupe de serveurs de noms appelés les serveurs racine.

Les serveurs de plus haut niveau sont de deux types : géographique et organisationnel. Les domaines géographiques, identifiés par deux lettres, représentent les pays (fr pour France, it pour Italie, jp pour Japon...).
Les domaines organisationnels sont originellement aux Etats-Unis :

- com : domaines commerciaux (beaucoup de sites non américains sont enregistrés sous ce domaine).
- edu : domaines de l'éducation et de la recherche.
- gov : domaines gouvernementaux.
- mil, net, int, org.

Les feuilles de cette arborescence désignent les machines.
Tous les domaines de plus haut niveau actuels sont gérés par l'organisme InterNic. Des propositions sont faites pour la création de nouveaux domaines de plus haut niveau. L'IAHC propose ainsi :

- firm : entreprises
- store : ventes
- web : services liés à W3
- arts : domaines liés à la culture
- rec : divertissements
- info : service d'informations

2.3.1 Création de domaines et de sous-domaines

L'organisme NIC alloue les domaines sous les domaines de plus haut niveau. Une fois cette autorisation accordée à un organisme, celui-ci est responsable du nommage de ses propres machines sans en avoir à en référer à une quelconque autorité. Il peut créer des domaines supplémentaires, appelés sous-domaines. Les enregistrements NS (Name Server) dans les bases DNS sont utilisés comme pointeurs. Quand un serveur local effectue une requête DNS, il conserve l'adresse et les pointeurs NS qui lui ont permis de la résoudre afin de s'en reservir pour d'éventuelles futures requêtes.

2.3.2 BIND, resolver et named

L'implantation du DNS utilisé sur les systèmes Unix est basée sur le programme BIND (Berkeley Internet Name Domain). Le programme qui joue le rôle de serveur DNS est divisé en deux composants : un résolveur et un serveur de noms.
Le résolveur est le programme qui crée la requête (implémenté sous la forme d'une bibliothèque de routines). Le serveur de noms, exécuté sous la forme d'un processus distinct appelé named, traite la requête. Avec BIND, tous les ordinateurs utilisent le code du résolveur mais pas forcément le processus de serveur de noms. Dans ce cas, la machine est appelée un système uniquement résolveur. Un serveur de noms peut être :

- primaire : Il contient les informations complètes concernant le domaine chargée à partir d'un fichier créé par l'administrateur du domaine.
- secondaire : Il transfère périodiquement la base de données du domaine (fichiers de zone) à partir du serveur primaire de manière à se mettre à jour. Il peut y avoir plusieurs serveurs secondaires.

Lorsque l'on ajoute une nouvelle machine dans le réseau, il est inutile de modifier les fichiers /etc/hosts des machines du réseau. Il suffit de modifier la base de données DNS sur le serveur primaire. Les informations sont alors automatiquement disséminées aux autres serveurs.

2.3.3 La commande nslookup

Elle permet, entre autre, de trouver l'adresse IP d'une machine en interrogeant le serveur de noms.
Exemple :

```
$ nslookup inria.inria.fr
Server :  lri.lri.fr
Address :  129.175.15.1

Non-authoritative answer :
Name :    inria.inria.fr
Address :  192.93.2.1
```

Traduction : nous avons demandé l'adresse IP de inria.inria.fr et le serveur de nom lri.lri.fr (dont l'adresse est 129.175.15.1) nous a répondu. L'adresse demandée est 192.93.2.1.

La commande nslookup lancée sans argument entre dans un mode interactif. Il est alors possible de spécifier le serveur DNS à interroger ou le type de réponse à afficher.
Exemple :

```
$ nslookup
Note :  nslookup is deprecated and may be removed from future releases.
Consider using the 'dig' or 'host' programs instead.  Run nslookup with
the '-sil[ent]' option to prevent this message from appearing.
> set type=any
> inria.fr
Server :         129.175.9.11
Address :        129.175.9.11#53
Non-authoritative answer :
inria.fr        nameserver = dns.inria.fr.
inria.fr        nameserver = ns2.nic.fr.
inria.fr        nameserver = nez-perce.inria.fr.
inria.fr        nameserver = imag.imag.fr.
inria.fr        nameserver = ns.eu.net.
inria.fr        nameserver = dns.princeton.edu.
inria.fr        nameserver = dns.cs.wisc.edu.
inria.fr        mail exchanger = 50 nez-perce.inria.fr.
```

```
inria.fr        mail exchanger = 50 concorde.inria.fr.
inria.fr
        origin = dns.inria.fr.
        mail addr = hostmaster.sophia.inria.fr.
        serial = 2002042502
        refresh = 21600
        retry = 3600
        expire = 3600000
        minimum = 7200
Authoritative answers can be found from :
inria.fr        nameserver = dns.inria.fr.
inria.fr        nameserver = ns2.nic.fr.
inria.fr        nameserver = nez-perce.inria.fr.
inria.fr        nameserver = imag.imag.fr.
inria.fr        nameserver = ns.eu.net.
inria.fr        nameserver = dns.princeton.edu.
inria.fr        nameserver = dns.cs.wisc.edu.
dns.inria.fr            internet address = 193.51.208.13
ns2.nic.fr              internet address = 192.93.0.4
nez-perce.inria.fr      internet address = 192.93.2.78
imag.imag.fr            internet address = 129.88.30.1
ns.eu.net               internet address = 192.16.202.11
dns.princeton.edu       internet address = 128.112.129.15
dns.cs.wisc.edu         internet address = 128.105.2.10
>
```

Ici c'est le serveur 192.175.9.11 qui nous a répondu, plus précisement un serveur qui écoute le port 53. La réponse ne fait pas foi (non-authoritative), c'est juste l'information contenue dans un cache qui nous a été transmise. Pour des réponses qui font foi, il faut interroger un des serveurs listés, comme par exemple dns.inria.fr, ns2.nic.fr, et on nous a aussi donné les numéros IP des ces serveurs. Ce qu'on sait sur le domaine inria.fr, c'est que le courrier doit être acheminé soit vers nez-perce soit vers concorde.inria.fr, chacun avec la même priorité. L'INRIA réparti donc la charge de manière équilibrée sur les deux serveurs SMTP. Plus loin on nous dit que cette information, le serveur 192.175.9.11 l'a obtenu directement du serveur dns.inria.fr, donc de première main. Si on a des questions, on peut contacter hostmaster@sophia.inria.fr (le premier point doit être remplacé par un @, le signe @ ayant déjà une autre signification dans les enregistrements DNS). Cette information porte un numéro de série, qui permet de décider en présence de deux enregistrements, lequel est le plus à jour. Le codage habituellement utilisé nous dit que c'est la deuxième mise à jour du 25 Avril 2002. D'une part c'est moins cher d'utiliser une information contenue dans un cache, d'autre part c'est en moyenne plus exact de demander les informations auprès des serveurs. Il y a donc un compromis à faire pour le choix des paramètres suivants. Le premier paramètre *refresh* indique la fréquence avec laquelle les serveurs secondaires doivent interroger le serveur primaire (ici tous les 6h). Si celui-ci ne répond pas, le serveur secondaire reéssaye tous les *retry* secondes (ici 1h), mais va quand même continuer à distribuer l'information pendant *expire* secondes (en général une semaine). Le temps *minimum* est le temps que ces enregistrements peuvent résider dans les caches, avant qu'il faille les jeter et redemander un enregistrement frais auprès des serveurs de noms (ici 2h).

2.4 NIS

NIS (Network Information Service) est un service utilisé pour administrer de façon centralisée les principales bases de données système d'un réseau de machines Unix. Cela concerne les fichiers comme passwd, group, hosts, negroup ...Les bases de données, appelées aussi *maps*, sont générées à partir de fichiers texte standards à l'aide d'un makefile utilisant la commande *makedbm* (les maps résultantes étant

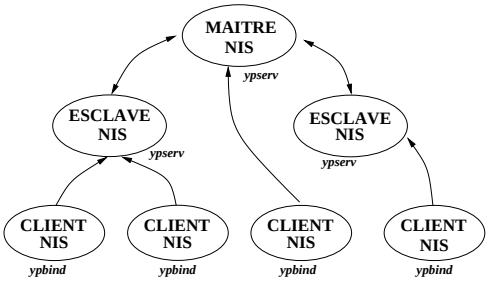

FIG. 2.2 – Diffusion des NIS

donc au format DBM).
Les maps NIS sont rangées dans le répertoire /var/yp/*nom_de_domaine* du serveur NIS et sont mises à la disposition des autres machines du domaine.
Il existe deux types de serveurs NIS :

- Le serveur maître, unique dans un domaine donné, contient et gère l'ensemble des maps NIS.
- Le serveur esclave est un serveur additionnel qui maintient l'ensemble des maps NIS en se mettant automatiquement à jour sur le serveur maître. Il peut y avoir plusieurs serveurs esclaves dans un domaine.

Les clients NIS exécutent le démon *ypbind* qui interroge le démon *ypserv* du serveur via les RPC. Le démon ypserv prend en charge la requête, consulte la map demandée et retourne l'information au client.

2.4.1 Installation d'un domaine NIS

2.4.1.1 Déclaration d'un serveur maître

La première étape est de créer un répertoire qui va héberger les fichiers ascii. Le répertoire par défaut est /etc. Il est fortement souhaitable de le changer pour que les fichiers qui s'y trouvent ne jouent pas à la fois le rôle de tables serveur et de tables client. Il convient de modifier le fichier */var/yp/Makefile* en initialisant la variable DIR au répertoire de son choix (par exemple /yp) et en copiant dans ce répertoire les différents fichiers destinés à être servis via les NIS. On peut éditer ces fichiers de façon à y supprimer les entrées que l'on ne désire pas (par exemple l'entrée root du fichier passwd).

Pour initialiser le serveur, il faut établir le domaine avec la commande *domainname* (ex : domainname IUT). Sur certains systèmes, il faut renseigner le fichier /etc/defaultdomain en y incluant le nom du domaine NIS. Il faut ensuite lancer la commande **ypinit -m**. Les bases de données DBM seront alors créées pour la première fois et rangées dans le répertoire /var/yp/*nom_de_domaine*. A chaque fois que l'on modifie un ou plusieurs des fichiers de /yp, il faut reconstruire les tables NIS à l'aide du Makefile (cd /var/yp ; make). Les maps ainsi générées sont propagées sur les esclaves avec la commande *yppush*. Les serveurs NIS doivent lancer au boot le démon ypserv (script /etc/rc.d/init.d/ypserv sous Linux).

2.4.1.2 Déclaration d'un serveur esclave

Tout comme pour le maître, il faut établir le domaine NIS avec la commande *domainname*. Il faut ensuite initialiser le serveur esclave en récupérant toutes les maps du serveur maître avec la commande *ypinit* : **ypinit -s *maitre-NIS***. Le serveur esclave, comme tout serveur NIS, doit lancer le démon *ypserv* au démarrage.

2.4.1.3 Déclaration d'un client

Les clients NIS doivent juste signaler qu'ils font partie du domaine NIS (commande *domainname*) et lancer le démon ypbind au démarrage. Dans ce cas, le démon est lancé en mode "broadcast" et c'est

le premier serveur NIS qui répond qui délivre les informations. Sur certains systèmes, il y a aussi une procédure d'initialisation du client avec la commande **ypinit -c**. Une liste de serveurs NIS est alors demandée et ce sont ces serveurs qui seront interrogés par ypbind (les adresses IP de ces serveurs doivent apparaitre dans le fichier /etc/hosts). A noter que les serveurs NIS sont aussi des clients et doivent à ce titre lancer le démon ypbind.

2.4.2 Les commandes NIS

- ypwhich : Cette commande permet de connaître le serveur NIS interrogé par le client NIS local.
- ypcat *mapname* : Cette commande permet d'afficher l'ensemble du contenu d'une map (ex : ypcat hosts).
- ypmatch *key* [*key* ...] *mapname* : cette commande permet de sélectionner les entrées d'une map correspondant aux arguments donnés (ex : ypmatch gus1 gus2 passwd).

2.4.3 Le fichier nsswitch.conf

Ce fichier définit l'ordre dans lequel les sources d'informations seront consultées. Toutes les bases de données gérées par NIS sont traitées dans ce fichier. On va y indiquer qui des fichiers locaux ou des NIS seront consultés en premier. Pour la résolution de noms de machines, le choix s'élargit avec le DNS.
Exemple d'un fichier nsswitch.conf :

```
passwd :     files nis
shadow :     files nis
group :      files nis
hosts :      files nis dns
services :   nis files
networks :   nis files
protocols :  nis files
rpc :        nis files
ethers :     nis files
netmasks :   nis files
bootparams : nis files
netgroup :   nis
publickey :  nis
automount :  nis files
aliases :    nis files
```

2.5 Telnet

Telnet est une application standard que fournissent la plupart des implémentations de TCP/IP et qui permet une connexion à distance sur une autre machine à travers le réseau par le biais d'un terminal virtuel. Il fonctionne entre des machines qui peuvent avoir des systèmes d'exploitation différents. Telnet utilise une négociation de l'option entre le client et le serveur pour déterminer quelles fonctionalités fournissent chaque extremité.

1. Le client telnet interagit à la fois avec l'utilisateur du terminal et avec les protocoles TCP/IP. Normalement tout ce que nous tapons est émis sur la connexion TCP et tout ce qui est reçu sur la connexion est affiché sur notre terminal.
2. Le serveur telnet fonctionne avec un driver de pseudo-terminal. Celui-ci fait croire au shell de login invoqué sur le serveur et à tout programme lancé à partir de ce shell qu'ils parlent à un terminal.
3. Une seule connexion TCP est utilisée. Le client telnet devant parfois parler au serveur telnet (et vice-versa), ils ont besoin de distinguer précisement toutes les commandes échangées sur la connexion par rapport aux données utilisateur.

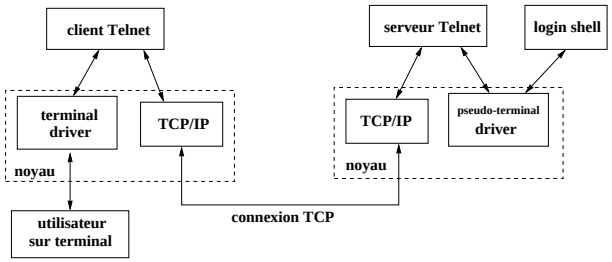

FIG. 2.3 – Application Telnet

4. Pour pouvoir ouvrir un shell sur la machine serveur, il faut bien sur y avoir un compte.

Exemple :
telnet psisun.u-psud.fr ou encore avec l'adresse IP :
telnet 193.55.10.132

```
newsun3%telnet 193.55.10.132
Trying 193.55.10.132...
Connected to 193.55.10.132.
Escape character is '^]'.

UNIX(r) System V Release 4. 0 (psisun)

login : gus
Password :
Last login : Thu Dec 15 8 :00 :39 from tdip1. u-psud. fr
Sun Microsystems Inc. SunOS 5. 3 Generic September 1993
#
....
```

2.6 FTP

C'est le plus ancien des services d'accès à des documents publics. Il repose très simplement sur l'utilisation de l'application FTP (File Transfert Protocol) qui permet de transférer des fichiers entre deux systèmes.

2.6.1 Principe de fonctionnement

FTP utilise deux connections TCP pour transférer des fichiers :

1. La connexion de contrôle est établie de façon normale en mode client-serveur. Le serveur fait une écoute passive sur le port bien connu 21 et attend une connexion d'un client. Le client fait une ouverture active du port 21 pour établir la connexion de contrôle qui reste active pendant toute la durée de la communication. Cette connexion est utilisée pour les commandes du client vers le serveur et pour les réponses du serveur. Le champs IP 'type de service' est du type 'minimise le délai' puisque les commandes sont normalement tapées à la main.
2. Une connexion de données est créée à chaque fois qu'un fichier est transféré entre le client et le serveur. Le champs IP 'type de service' est du type 'maximise le débit' puisque cette connexion est réservée au transfert de fichiers.

Une utilisation intéressante de cette commande réside dans les accès anonymes. Elle permet aux utilisateurs de transférer des fichiers depuis une machine sur laquelle ils ne sont pas enregistrés. La connexion en anonyme sur un site qui autorise ce type de connexion, se réalise simplement en entrant **anonymous ou ftp** comme nom de login et son e-mail en guise de mot de passe. Cette phase de connexion terminée, on peut se déplacer dans l'arborescence (cd), visualiser le contenu du répertoire (ls) et rapatrier des fichiers (get).

Les transferts de fichiers s'effectuent en mode ascii (commande ascii) pour les fichiers textes ou en mode binaire (commande bin) pour les autres. La liste des commandes à disposition est restreinte et peut s'obtenir par un ?. D'autres clients ftp plus conviviaux existent sous X-Windows : par exemple xftp gftp, ftptool et Netscape.
Tous les produits domaine public (X11, TEX, applications GNU ...) sont disponibles par ftp.

2.7 Sendmail

Le démon sendmail du sytème Unix joue le rôle d'agent transporteur et d'agent distributeur pour le réseau TCP/IP dans lequel il dialogue au moyen du protocole SMTP (Simple Mail Transport Protocol). Sendmail utilise les informations contenues dans les fichiers aliases et .forward. Sendmail contrôle les messages, interprète les adresses des destinataires, choisit l'agent distributeur approprié, réécrit les adresses dans la forme interprétable par l'agent ditributeur, reformate les entêtes de messages vers l'agent distributeur. Sendmail génère des enregistrements gérés par le démon syslogd. Les actions de sendmail sont paramétrées au moyen du fichier de config sendmail.cf (ce fichier est différent selon qu'il s'agit d'une machine relai ou d'une machine ordinaire).
Ce fichier comprend trois parties :

1. la définition de paramètres propres au site
2. les règles de réécriture des adresses
3. le choix des agents distributeurs

2.7.1 Lancement de sendmail

Le démon sendmail est généralement lancé au demarrage de la machine. Le script de démarrage de sendmail sous Linux est /etc/rc.d/rcx.d/S80sendmail. Sendmail utilise le port bien connu 25.

2.7.2 Le fichier aliases

Ce fichier contient des listes d'utilisateurs et des surnoms. Le format des alias est le suivant :

Alias : destinataire[,destinataire,...]

Exemple :

```
Mdf : Michel de fronsac
Equipe-bd : leonard, mathilde, emmanuel, vero, nicolas
```

On peut gérer des listes de discussion.
Exemple :

```
Owner-equipe-bd : pierre
Equipe-bd-request : pierre
```

Sendmail n'utilise pas directement le fichier aliases. Ce fichier doit être traité par newaliases qui va créer les bases de données utilisées par sendmail.

2.7.3 Le fichier .forward

Ce fichier permet à un utilisateur de définir son propre renvoi de courrier. Il se trouve dans le répertoire HOME et contient la liste des adresses à qui on veut renvoyer les messages reçus.

Chapitre 3

Les parefeux

Les documentations du *Linux Documentation Project* concernant ce sujet sont

1. le firewall-HOWTO, qui n'est pas très technique et ne traite pas la commande iptables,
2. le chapitre 9 du *Linux Network Administration Guide version 1.1*, qui est assez lisible mais ne va pas dans tous les détails,
3. le document iptables-HOWTO écrit par l'auteur de iptables, qui est très bien écrit.

Dans la vraie vie un parefeu est un mur en briques séparant plusieurs parties d'un bâtiment qui bloque la propagation d'un feu. Dans une voiture c'est une plaque en métal entre le moteur et la cabine des passagers qui joue le même rôle. En informatique il s'agit d'un routeur entre Internet et un réseau local qui est doté de règles spécifiant quels sont les paquets IP qui sont autorisés à traverser le routeur.

Un parefeu a plusieurs buts.

1. Le but le plus important est de protéger le réseau local d'attaques venues d'Internet.
2. Parfois on installe un parefeu pour restreindre les utilisateurs du réseau local à certains services d'Internet seulement. C'est le cas à l'IUT actuellement, où les étudiants peuvent seulement utiliser le web, ftp et le courrier.
3. Un parefeu peut aussi servir à espionner les utilisateurs du réseau local. Il peut être configuré afin de laisser passer tous les paquets, mais de garder des traces de certains paquets bien spécifiés.

Quand on installe un parefeu on doit définir une politique de sécurité. Quel groupe d'utilisateurs doit avoir accès à quels services. D'une part on veut protéger le réseau local, d'autre part on veut que les gens puissent travailler confortablement.

3.1 Les types d'attaque

Voici quelques types d'attaque et comment y remédier.

accès non-autorisé Une personne accède à un service d'une machine qu'elle n'est pas censée accéder. Il faut autoriser seulement un groupe de personnes bien définis.

exploitation de failles connues Certains programmes ont été écrit dans un environnement de confiance et comportent de nombreuses failles. Par exemple certains systèmes d'exploitation étaient livrés avec un utilisateur prédéfini, pour permettre un accès à distance pour la maintenance. Or ceci était inconnu de la plupart des administrateurs qui ne changeaient pas le mot de passe de cet utilisateur. Il faut désactiver les services aux failles connues, dont la liste est mis à jour régulièrement à security.web.cern.ch. On peut alors installer des versions plus récents de ces programmes ou trouver des alternatives. Par exemple il est conseillé d'utiliser ssh plutôt que rsh, logiciel permettant d'exécuter une commande ou un shell à distance.

déni de service (denial of service) Il s'agit d'attaquer une machine de telle manière à ce qu'elle ne puisse plus rendre les services qu'elle est sensée rendre. Par exemple une machine est bombardé de demandes de connexions, qui ne vont jamais aboutir, et le serveur est obligé d'allouer des ressources inutiles pour chaque demande. Au bout d'un moment il tellement occupé avec ceci (et peut être commence-t-il a swapper) qu'il ne peut plus rendre les autres services. On se souviendra de l'attaque contre les serveurs de yahoo, cnn, buy et amazon en février 2000, d'un étudiant de Santha Fe. Pour se parer ce ces attaques, il faudrait filtrer les paquets suspects (p.ex. dont l'adresse expéditeur semble faussée) et de n'autoriser qu'un nombre limité d'accès par seconde (ce qui peut être spécifié avec xinetd).

spoofing Il s'agit de la génération de paquets en forgeant les entêtes de telle manière à ce que les paquets s'insèrent dans une connexion déjà existante, pour profiter des privilèges de celle-ci. Il faudrait vérifier l'authenticité des paquets (ce qui est p.ex. possible en IP version 6) et de ne pas routeur les paquets si l'adresse expéditeur semble invalide. Cette dernière mesure protège le reste du monde d'attaques venues du réseau local, mais si tout le monde suivrait cette mesure, tout le monde serait un peu plus protégé.

espionnage C'est l'attaque la plus simple à mettre en oeuvre. Certains logiciels (comme etherreal p.ex.) permettent de capter le trafic sur le réseau local, et ainsi de récupérer les mots de passe, afin d'obtenir par la suite un accès non-autorisé. L'usage de commutateurs au lieu de hubs, limite ce type d'attaque. Beaucoup de trafic réseau peut être chiffré, voir ssh, https, etc.

3.2 L'architecture des parefeux

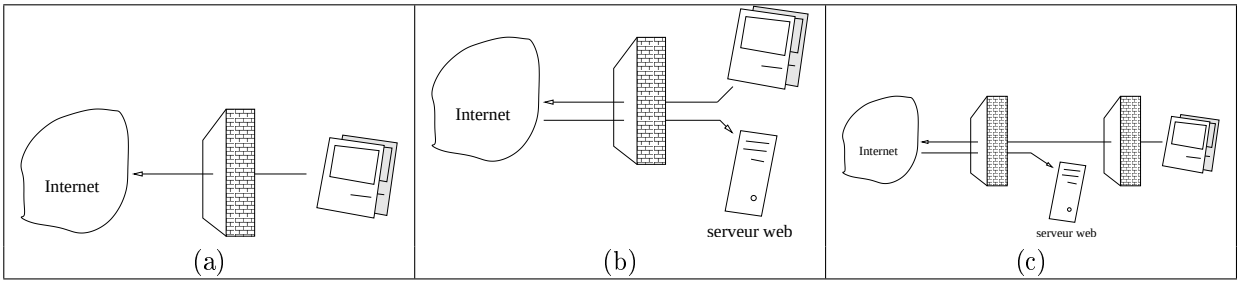

FIG. 3.1 – Différents architectures des parefeux

La figure 3.1 montre une liste non-exhaustive de différents architectures de parefeu. Le plus simple est la forme 3.1a, ou un parefeu — donc un routeur entre le réseau local à droite et Internet à gauche — ne laisse passer que des demandes de connexions vers Internet et pas dans l'autre direction. Par contre des paquets IP d'une connexion déjà établie peuvent (et doivent) circuler dans les deux sens. Cette situation pourrait à l'extrême se réduire à une seule machine connectée à Internet, et ne permettant aucune connexion sur celle-ci. Un des problème étant bien sûr que le courrier ne peut plus être déposé sur le serveur smtp du réseau local, et en général ce réseau ne peut fournir plus aucun service. Dans la figure 3.1b on autorise les connexions du reste du monde vers une machine particulière du réseau local, sur lequel tourneront les différents serveurs, serveur web par exemple. Un problème de cette architecture est que si quelqu'un prend possession du serveur il peut de là attaquer les autres machines du réseau local. La situation 3.1c s'en protège en définissant deux zones de sécurité distinctes. Dans la première se trouvent les serveurs, et dans la deuxième les autres machines du réseau local. Ainsi seul le premier parefeu laissera entrer des connexions vers le serveur, et du serveur il est à priori impossible de se connecter vers la deuxième zone.

3.3 La commande iptables

Suivant les types de routeurs, les règles de filtrage du parefeu se configurent de manière différentes. Nous considérons comme routeur une machine sous Linux avec plusieurs cartes réseau. Entre parenthèses pour qu'elle accepte de routeur les paquets, il faut mettre de façon permanent à 1 la variable système net.ipv4.ip_forward en éditant le fichier /etc/sysctl.conf.

Sous Linux il y a trois systèmes de parefeu : ipfwadm, qui est le tout premier système, ipchains, qui est plus récent et iptables qui remplace les deux derniers. Le modèle de iptables le rend plus simple à utiliser. Iptables fait partie de tous les noyaux récents de Linux.

Les règles de parfeu sont écrits dans la mémoire du système. Et tout comme la table de routage peut être lue et écrite par la commande route, ces règles sont accessibles avec la commande iptables.

Pour iptables, il y a des *tables*, dans les tables il y a des *chaînes* et dans les chaînes des *règles*. La table se spécifie par -t et peut être *mangle*, *nat* ou *filter*, et c'est cette dernière qui est utilisée par défaut si rien n'est spécifié. C'est ce que nous supposons par la suite.

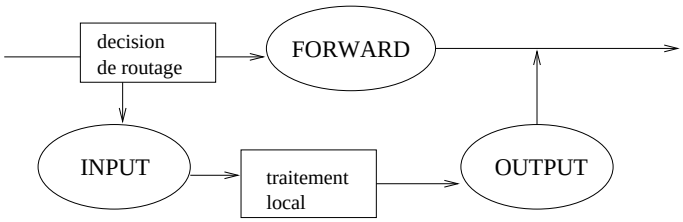

FIG. 3.2 – Les trois chaînes dans iptables

Il y a trois chaînes FORWARD, INPUT, OUTPUT, voir figure 3.2. Lorsqu'un paquet IP arrive de l'extérieur sur la machine, il traverse la chaîne INPUT si le destinataire est la machine elle-même, et la chaîne FORWARD sinon. La chaîne OUTPUT est traversée par les paquets émis de la machine vers l'extérieur.[1] Qu'est-ce que ça veut dire qu'un paquet IP traverse une chaîne ? Une chaîne a une politique par défaut et une liste de règles. Une règle associe une action à une spécification de paquets. Par exemple (-p icmp -j ACCEPT) va accepter tous les paquets comportant des messages ICMP, par exemple ceux générés par la commande ping. Donc le paquet est comparé à toutes les règles de la chaîne, un par un, et dés qu'une règle est trouvée qui correspond à ce paquet l'action correspondante est déclenchée. Si aucune règle ne correspond, c'est l'action par défaut (politique par défaut) de la chaîne qui est déclenchée.

Une action peut être ACCEPT, qui va faire passer le paquet, DROP, qui va le détruire, et RETURN qui va en plus envoyer un message à l'expediteur pour l'informer que le paquet a été détruit. L'action d'une règle est spécifiée par l'option -j.

3.4 Modification des chaînes

Une chaîne est vidée de toutes ses règles par l'option -F (comme flush). -F tout seul vide toutes les chaînes. On peut ensuite ajouter des règles avec -A (comme add). La politique par défaut est modifiée avec -P.

Exemple :
iptables -F INPUT
iptables -P INPUT DROP

Ces instructions ne laissent plus passer aucun paquet vers la machine elle-même. Notez que non seulement on ne pourra plus accéder à ses services, mais de la machine même on ne pourra plus se connecter vers

[1] Je crois que les paquets émis par un processus de la machine à un autre processus de la même machine ne soient pas filtrés du tout, mais Pierre Boulan dit qu'ils passent par les trois chaînes. Je n'ai pas trouvé la réponse dans la documentation.

d'autres machines, car le retour ne passera plus. D'ailleurs elle ne recevra même plus les demandes d'écho par ping.

3.5 Spécification des paquets dans iptables

Pour spécifier les paquets IP provenant d'une interface particulière ou qui vont sortir sur une interface particulière, on utilisera les options `--in-interface ou --out-interface`.

L'adresse IP source et destination peut être spécifiée à l'aide des options `--source` et `--destination`.

Exemple :
iptables -F FORWARD
iptables -P FORWARD DROP
iptables `-A FORWARD` --source 192.168.30.32/30 -j ACCEPT
iptables `-A FORWARD` --destination 192.168.30.32/30 -j ACCEPT

Cet exemple laissera passer tous les paquets vers (ou issues des) adresses 192.168.30.32 à 192.168.30.35, et seulement ceux-la.

Le protocole au dessus de IP peut être spécifié par `-p`, par exemple `-p icmp` ou `-p udp` ou `-p tcp`. Dans les deux derniers cas on peut en plus spécifier le port source et destination avec les options `--source-port` et `--destination-port`. Le port peut soit être un numéro, soit un nom de service, tel qu'on le trouve dans /etc/services. Dans le cas de TCP on peut spécifier les paquets comportant une demande de connexion avec l'option `--syn`.

La plupart des options existent en version longue (avec deux tirets) et en version courte (souvent avec un seul tiret). Par exemple `--source-port` et `-s` sont identiques. La page manuel de iptables en dit plus.

Exemple :
iptables -F FORWARD
iptables -P FORWARD ACCEPT
iptables -A FORWARD -i ppp0 -p tcp --source-port ! 80 -j DROP

Cette exemple accepte tous les paquets sauf les demandes de connexion venant de la connexion Internet ppp0 vers un serveur, autre qu'un serveur web. Pour être précis il faudrait filtrer aussi le protocole UDP.

3.6 Les compteurs

Iptables associe à chaque règle un compteur de paquets et d'octets. On peut alors spécifier une règle sans action, juste pour avoir un compteur, comptabilisant le trafic d'un certain type spécifié. Ceci permet de mesurer la part du web dans le trafic réseau, ou de partager les coûts d'une connexion Internet proportionnels à l'utilisation faite par gens qui la partagent. L'option -L relève les compteurs, l'option -Z les remet à zéro.

3.7 Le problème des protocoles à port dynamique

Prenant le cas de figure 3.1a d'un parefeu ne laissant passer que des demandes de connexion du réseau local vers Internet et non l'inverse. Le problème est que certains protocoles ont besoin d'une deuxième connexion émanant du serveur vers le client. Dans FTP par exemple, le client se connecte sur le serveur sur le port 21, ce qui établie un canal par lequel transitent les commandes de liste de répertoire ou de rapatriement d'un fichier p.ex. Ensuite pour transmettre un fichier, le client demande au système un port encore libre, qui lui sera attribué dynamiquement et envoie ce numéro au serveur. Celui ci s'y connecte et le transfert de fichier est effectué à travers ce canal de données, voir figure 3.3(actif). Le fait d'avoir un canal séparé permet par exemple d'interrompre le transfert plus facilement, et aussi il ne faut pas un mécanisme particulier pour distinguer dans un flux unique les données des commandes, comme dans smtp ou http.

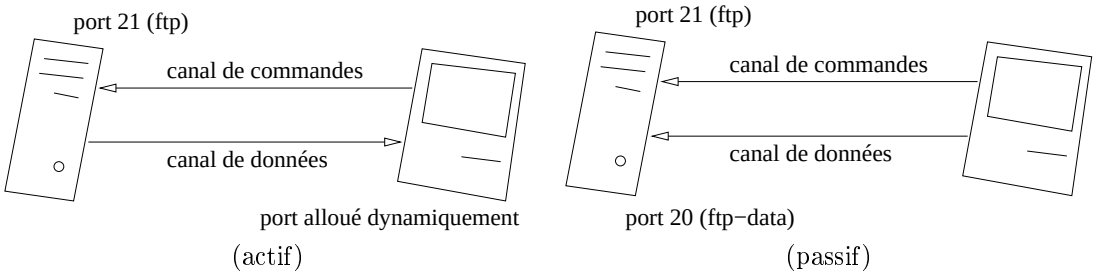

FIG. 3.3 – Les deux modes de FTP.

Le problème est que le numéro de port qu'écoute le client peut être n'importe quoi, car il est attribué dynamiquement, mais que nous ne voulons pas permettre au parefeu de laisser passer toutes les demandes de connexions. Il y a alors deux solutions.

1. On peut utiliser le mode passif de ftp. Ici c'est le client qui ouvre le canal de données en se connectant sur le port 20 (ftp-data) du serveur. Bien que les deux schémas semblent très semblables, ce dernier mode est plus difficile à mettre en oeuvre côté serveur, car il faut lors d'une demande de connexion sur le port 20, trouver le processus qui est en train de parler avec ce même client pour lui passer les données, voir figure 3.3(passif). Par contre il suffit sur le parefeu d'autoriser les connexions sortantes pour le port destinataire 20 et 21. Le client ftp de netscape implémente ce mode par exemple.
2. Pas tous les protocoles à port dynamique ont un tel mode passif. IRC (Internet Relay Chat) en est un exemple. On peut alors, sur le parefeu, mettre en place un mécanisme qui analyse les connexions en cours et ne laisse passer que les demandes de connexions entrantes si elles sont reliées à une connexion en cours sortante. Il faut charger le module state dans iptables et la commande devrait ressembler à quelque chose comme

   ```
   iptables -A INPUT -m state --state RELATED -i ppp0 -j ACCEPT
   ```

3.8 Traduction d'adresses réseau (ou NAT, IP masquerading)

Voici une situation courante. Un réseau local utilisant des adresses IP privées possède une machine, qui elle est connecté à Internet, et comporte sur cette connexion une adresse IP valide (non-privée) qui peut statique ou dynamique, peu importe. Cette machine (appelons la *serveur*) veut faire partager la connexion Internet aux autres machines du réseau local. Le problème est que les adresses privées ne doivent pas sortir vers Internet. D'ailleurs un paquet avec une adresse destinataire privée n'y trouvera pas sa destination. Alors plusieurs solutions sont possible. (Il faut toujours proposer plusieurs solutions, pour laisser croire aux gens qu'ils aient le choix.)

1. Les clients se connectent sur le serveur et de la vers Internet. Par exemple

   ```
   host1> telnet serveur
   login :...
   serveur> export DISPLAY=host1 :0
   serveur> netscape &
   ```

   Le problème avec cette solution est qu'elle n'est pas pratique pour les utilisateurs et fonctionne mal en milieu hétérogène comportant des machines sous MacOS, Microsoft Windows, Unix, etc.
2. On installe sur le serveur un serveur proxy pour les services que les clients prévoient d'utiliser. Un serveur proxy (web par exemple) se comporte comme un serveur web pour les clients, et comme un client pour les véritables serveurs web. Les clients doivent configurer leurs applications (netscape par exemple) afin d'utiliser ce serveur proxy pour les demandes de service, qui lui se chargera de les

fournir auprès des véritables serveurs. Dans ce cas aucune connexion TCP ne traverse le serveur, il y aura deux connexions distinctes. Le problème est qu'il faut configurer les clients, prévoir comment ils vont utiliser Internet, et le serveur sera peut être chargé avec le travail de proxy.

3. La solution la plus transparente pour les clients est la traduction d'adresses réseau, NAT pour Network Address Translation en anglais ou IP masquerading. L'idée est de mettre en place sur le serveur un mécanisme qui lors du routage de paquets IP du réseau local vers Internet, remplace l'adresse expéditeur (privée) par la sienne (valide). Lorsque des paquets IP reviennent vers le serveur il effectue l'opération inverse. Afin de retrouver l'adresse du véritable destinataire, le remplacement est fait sur l'adresse IP et le numéro de port. Dans iptables ceci est réalisé de la manière suivante.

   iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

   L'idée est que iptables a dans la table NAT une chaîne POSTROUTING qui est traversée après toutes les autres chaînes de la table filter. L'action MASQUERADE va alors changer l'adresse IP et le port source pour les paquets sortants et l'adresse IP et le port destination pour les paquets entrants, voir figure 3.4.
   Le résultat de cette opération est que les clients peuvent se connecter à Internet de manière complètement transparente, et apparaîtront comme une unique machine (le serveur) de l'extérieur. Par contre des machines dans Internet ne peuvent pas ouvrir directement une connexion avec les machines locales.

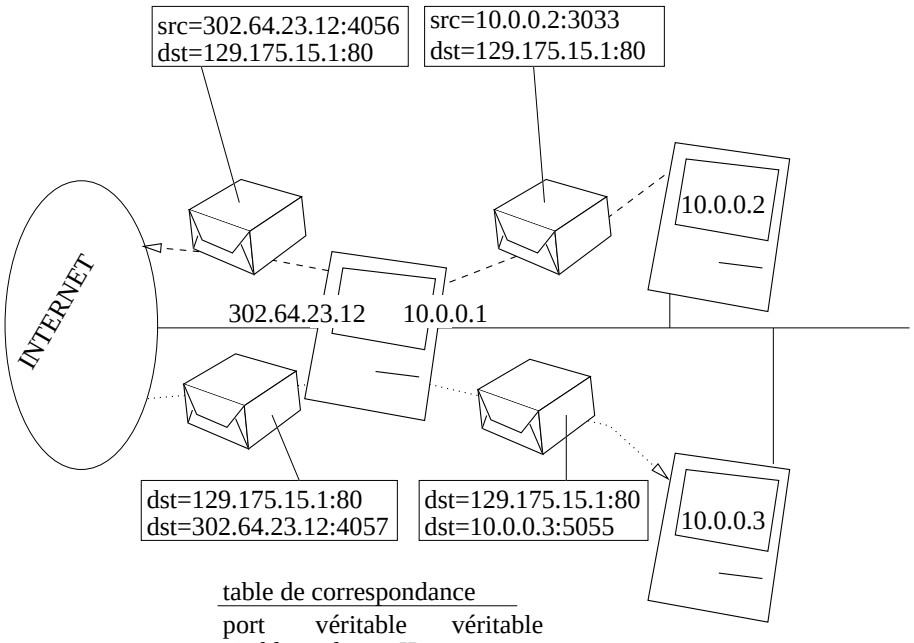

table de correspondance

port visible	véritable adresse IP	véritable port
4056	10.0.0.2	3033
4057	10.0.0.3	5055

FIG. 3.4 – La traduction d'adresses réseau

Entre parenthèses il existe un mécanisme inverse à la traduction d'adresses qui s'appelle RNAT et ou de nouvelles entrées sont faits dans la table de correspondance lors de demande de connexion *de l'extérieur vers le réseau local* cette fois ci. Dans iptables ceci est fait par la chaîne PREROUTING, et ceci permet de répartir la charge d'un serveur web par exemple sur plusieurs serveurs. Je soupçonne google de faire ainsi. Ceci est une alternative à la distribution de charge par DNS, qui associe plusieurs numéros IP à un unique nom comme www.cnn.com.

3.9 En guise de résumé

	parefeu	xinetd	application
adresse physique	X		
adresse IP	X	X	
login/mot de passe			X

FIG. 3.5 – Des critères d'autentification effectués à différents niveaux.

Permettre ou ne pas permettre l'utilisation d'un service peut en résumé se décide à différents niveaux. Un parefeu peut filtrer les paquets en fonction du numéro IP, et en utilisant un module d'iptables, peut filtrer en fonction des adresses physiques. Ceci est intéressant, car les adresses IP sont falsifiables, mais pas vraiment les adresses physiques. Par exemple dans un réseau local, on veut pouvoir intervenir à distance sur un parefeu, mais seulement à partir de la machine de l'administrateur réseau. Une fois que les paquets ont traversés le parefeu ils sont accueillis en général par xinetd sur la machine, processus central, qui lance les services demandés quand il le faut. Dans ses fichiers de configuration (dans /etc/xinetd/) on peut spécifier quelle machine à le droit d'utiliser quel service. Finalement, une fois la connexion est établie entre client et serveur, ce dernier peut demander par exemple login et mot de passe pour être sûr que le service est rendu à une personne autorisée. Souvenez vous, quand vous essayez de vous connecter sur www.lri.fr/~durr/prive/ un login/mot de passe est demandé. La figure 3.5 synthétise de manière simplifiée ceci.

Chapitre 4

Les réseaux publics

Un réseau local est un réseau à l'intérieur d'un immeuble. Par contre, pour connecter deux ordinateurs qui sont dans des immeubles différents, on est obligé de recourir aux services de l'opérateur de télécommunication national.

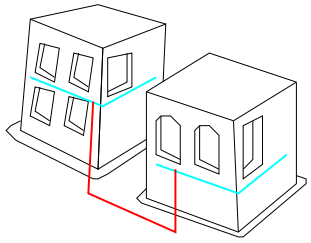

FIG. 4.1 – En général la loi impose que deux réseaux locaux séparés par une route publique doivent être connectés par l'opérateur de télécommunication national. (En France c'est Oléane, une filiale de France Télécom.)

Le débit est environ de 10 à 100 Mbits/s avec un taux d'erreur immesurablement petit (1 erreur par jour peut-être). Si on utilise le réseau téléphonique commuté (RTC), le débit est environ de 10 kbits/s avec un taux d'erreur important.

	Débit	Taux d'erreurs
local	10-100 Mbits/s	10^{-12} ou 10^{-13}
RTC	10 kbits/s	10^{-5}

Il y a donc eu besoin de réseaux publics pour la transmission de données. On pourrait louer des liaisons spécialisées, mais en général c'est une solution trop couteuse.

4.1 Rapide Histoire

A ma connaissance, le premier réseau était utilisé dans l'antiquité en Israël et il permettait de transmettre un seul bit. Les différents maîtres de cérémonie juifs devaient être informés de la date exacte du début d'une fête religieuse. Le jour en question le maître chef mettait un feu en haut d'une colline. Puis les autres maîtres, dès qu'ils voyaient le feu, en mettaient un à leur tour sur leur colline. Ainsi le bit d'information se propeagait dans le pays.

Plus tard au début du 19-ème siècle la France avait mis en place un réseau de télégraphes aériens. Ce sont des tours munies de bras articulés installées sur des collines bien dégagées. Un message de Strasbourg à Paris mettait six heures. Souvent c'est sur ces collines qu'on trouve aujourd'hui les émetteurs radio.

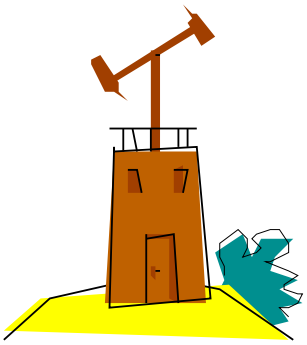

FIG. 4.2 – télégraphe aérien

Comme ce réseau était bien établi, la France était un peu en retard sur le télégraphe électrique. La situation similaire s'est reproduite avec le minitel et Internet en France. Depuis 1988 le réseau téléphonique en France est digital.

On va maintenant énumérer différents réseaux publics.

4.2 SMDS - Switched Multimegabit Data Service

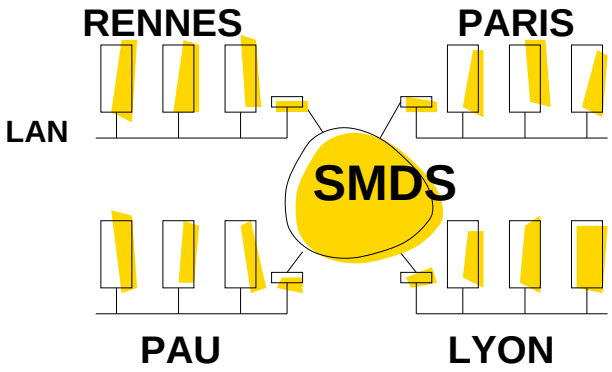

FIG. 4.3 – SMDS permet de connecter n machines avec n liaisons courtes machine-réseau plutôt qu'avec $n(n-1)/2$ liaisons longues machine-machine.

- connecter les LAN à SMDS par liaison louée par exemple.
- SMDS joue le rôle d'épine dorsale.
- débit standard de 45 Mbits/s
- est un réseau commuté (contrairement aux MAN (Metropolitan Area Network))
- filtrage entrant et sortant sur les adresses destination et source (pour les réseaux de données confidentielles)

Adresse Destination	Adresse source	Données (peut être IP, Ethernet, peu importe)

FIG. 4.4 – Une trame SMDS

4.2.1 Gestion pointe de traffic

Le routeur de connexion au réseau SMDS a un compteur qui est incrémenté toutes les (disons) 10 μs. Le paquet arrive : si longueur>compteur alors le paquet est détruit sinon le paquet est envoyé et compteur -= longueur.

Un top d'horloge toutes les 10 μs donne un débit moyen de 100000 octets/s. Par contre si pendant 10 ms, il n'y a pas eu d'émission, le compteur est à 1000. Conclusion : On peut donc envoyer 1 Ko à 45 Mbits/s en environ 180 μs.

Si on avait une liaison louée de 100 000 octets/s, il aurait fallu 10 ms. Ce mécanisme permet à l'utilisateur d'avoir des rafalles de trafic important, donc un bon temps de réponse pour des applications interactives, et permet à l'opérateur d'assurer que le débit moyen reste faible.

4.3 MAN - Metropolitan Area Network

MAN est d'un point de vue technologique un grand LAN.
- couvre une ville, campus, public ou privé.
- voix, ou/et données, éventuellement relié à un réseau câblé de télévision.
- sans commutateur, 1 ou 2 câbles.

Un MAN utilise la technologie..

4.3.1 DQDB - Distributed Queue Dual Bus

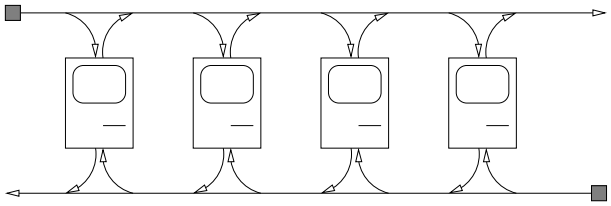

FIG. 4.5 – DQDB est composé de deux bus unidirectionels auquels sont connectées les machines. Les extrémités des bus disposent d'un générateur de trames vides.

Les trames DQDB (appelées cellules) ont une taille unique de 53 octets avec 44 octets de charge utile (compatible avec ATM) et 2 bits particuliers à DQDB :
- busy (indique si la cellule est vide ou utilisée,)
- et request.

Chaque ordinateur doit avoir comme connaissance du réseau qui est à gauche et qui est à droite. Pour envoyer des données il suffit de les mettre dans une cellule vide qui passe sous le nez. On pourrait imaginer que les machines aux extrémités ne cessent d'envoyer des données, ce qui empècherait les autres de parler. Il faut donc une gestion d'accès au support, ce qui réalisé pour chaque bus par une..

4.3.1.1 File d'attente décentralisée

Priorité à ceux qui sont en aval (loin de la tête du bus). Supposons que tout le monde ne veut envoyer des données que vers la droite. Pour cela on va envoyer les données sur le bus du haut, et on va utiliser le bus du bas pour demander aux voisins de gauche de nous laisser une cellule vide.

Chaque machine a deux compteurs RC (request counter) et CD (countdown). Si une machine veut envoyer elle met à 1 le bit request d'une cellule qui passe sur le bus du bas et met RC dans CD et RC à 0. Elle envoie dès que CD est 0. Voici comment CD peut devenir zéro.

RC est incrémenté à chaque fois qu'on voit passer le bit request à 1.

CD est incrémenté à chaque passage d'une cellule vide.

Malheuresement cette technologie n'a pas eu beaucoup de succès. Les opérateurs préféraient investir dans le futur réseau ATM. Par contre, les Etats-Unis, Allemagne, Italie, Australie ont pris DQDB en attendant.

4.4 X.25 - nom commercial en France : Transpac

Vous connaissez Transpac, c'est le réseau utilisé par le minitel. La couche liaison est équivalente à HDLC. Existe depuis 1970.

Couche réseau : adressage, contrôle de flux, confirmation de remises, établissement de circuit virtuel (en mode commuté ou permanent), paquets de taille ≤ 128 octets, transmission fiable, préserve le séquencement.

4.5 Relais de trames

C'est une connexion entre deux machines qui utilise un réseau (en général ATM) mais simule une ligne louée.

- ligne louée : relais de trames peut avoir des raffales de débit plus important pour le même prix.
- X.25 : relais de trames propose moins de services. Trames erronées sont détruites. C'est aux utilisateurs de s'occuper de la reprise sur erreur.

4.6 Comparaison des services

Fonctionalité	DQDB	SMDS	X.25	Rel. de tr.	ATM
Orienté connexion	oui	non	oui	oui	oui
Débit normal (Mbit/s)	45	45	0,064	1,5	155
commuté	non	oui	oui	non	oui
Format fixe	oui	non	non	non	non
Taille max. (octets)	44	9 188	128	1 600	variable
CV permanents	non	non	oui	oui	oui
Multidistribution	non	oui	non	non	oui

4.7 ATM

Asynchronous Transfer Mode ("Asynchronous" par opposition à SONET (Synchronous Optical Network) du Téléphone longue distance depuis 1980)

La coexistance des nombreux réseaux différents est sans doute dûe à l'éclatement de AT&T en 1984.

ATM est un projet de réseau haut débit de l'industrie de télécommunication et l'industrie informatique.

- Faire un réseau unique pour téléphone, télex, télévision, ordinateurs.

4.7.1 En deux mots

- Orienté connexion (virtuelle)
- Commutation de cellules (paquet de taille unique et petite)

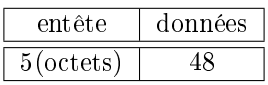

FIG. 4.6 – une cellule ATM

- débit 155,52 Mbits/s (nécessaire pour la diffusion de télévision)
- déjà utilisé :
  - Rénater (Réseau national de l'enseignement et de la recherche).
  - Liaisons louées (sont maintenant que des tronçons d'un réseau ATM)
  - Le Chili à tout son réseau téléphonique en ATM.
- Changement énorme pour les opérateurs de télécoms
  - Remplacer une bonne partie du réseau téléphonique
  - Rompre avec la tradition de la communication par circuits.
  - Qui va payer les frais ? Est-ce que la vidéo à la demande va pouvoir concurencer avec la location de cassettes vidéo ?

4.7.2 Organisation de ATM

couche OSI	couche ATM	sous-couche ATM	Fonctionalitées
3/4	AAL	CS	Fournir une interface standard (convergence)
		SAR	Segmentation et réassemblage
2/3	ATM		Contrôle de flux, Génération/extraction entête des cellules, Gestion des conduits/voies virtuels, Multiplexage/démultiplexage des cellules
2	physique	TC	Découplage débit cellule, Génération et vérif. total contrôle en-tête, Empaquetage et dépaquetage des cellules, Génération de trames
1		PMD	Synchro bit, Accès physique au réseau

AAL ATM Adaptation Layer

CS (Convergence Sublayer) : S'occupe de décomposer des paquets en cellules et vice-versa.

SAR (Segmentation and Reassembly) : Propose différents services de manière à utiliser ATM conviviale-ment.

ATM n'est pas divisé en sous-couches. Mélange services liaison et réseau.

TC (Transmission Convergence) : Emission : Transfert de cellules en chaînes de bits et envoie.
Reception : Trouver le début de cellules
Dans ATM, ceci est appelé couche physique mais dans OSI et ailleurs, c'est appelé couche liaison.

PMD (Physical Medium) : va être spécifique à Ethernet, FDDI, etc.

4.7.3 AAL

4.7.3.1 Différents modes de services

- orienté connexion (transmission par flot) ou sans connexion (transmission de messages)
- temps réel ou temps non réel
- débit constant ou débit variable

Parmi les huit combinaisons possibles, seules quatre ont été retenues comme utile. L'UIT a défini les protocoles AAL1 à AAL4. Quand l'industrie a vu ce qui a été proposé (par exemple 8% de contrôle dans la communication), ils ont défini un autre protocole, le SEAL (Simple Efficient Application Layer), adopté par l'UIT sous le nom de AAL5.

SEAL permet au choix une connexion unicast, multicast, un service fiable, non fiable, une transmission par flux ou par messages. Les données sont transmises sous la forme suivante.

charge utile	uu	(réservé)	length	CRC
1 à 65535 octets	1	1	2	4

uu user to user pour couche supérieure

length de charge utile sans octets de bourrage (pour arriver à un multiple de 48 octets) (on peut mettre 0 pour avorter un message en cours)

CRC code cyclique

La sous-couche SAR découpe en cellules et demande à ATM de mettre le bit PTI de la dernière cellule à 1 pour préserver la frontière des messages (violation du principe de l'organisation en couches des réseaux).

4.7.4 ATM - Couche Réseau

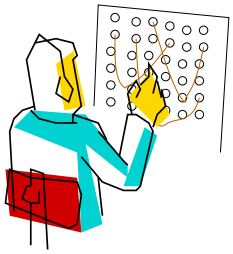

FIG. 4.7 – Les premiers commutateurs

Au milieu du 19-ème siècle, le téléphone a été inventé, on reliait les téléphones entre eux grâce à un câble, il y avait une opératrice qui s'occupait de relier les deux interlocuteurs en les mettant en contact grâce à un câble.

Petite anecdote : Dans une petite ville, il y avait 2 services de pompes funèbres et l'un des directeurs de celles-ci, avait une femme opératrice, ainsi quand un décès avait lieu, c'était toujours le même qui était contacté. Pour ne pas faire faillite, l'autre a été "contraint" d'inventer le commutateur...

En gros pour transmettre sur ATM il faut..

- Ouvrir un circuit virtuel (en envoyant une demande sur le circuit virtuel (CV) numéro 0 avec l'adresse de destination) va nous donner un numéro de CV.
- utiliser ce numéro pour envoyer des cellules.

Contrairement à IP, les cellules ne contiennent pas l'adresse destinataire. Ils contiennent seulement un numéro de CV.

numéro CV	...	HEC	charge utile
28 bits		1 octet	48 octets

FIG. 4.8 – Détail d'une cellule ATM (HEC = header error correction)

Chaque noeud du réseau ATM a des entrées et des sorties. Son rôle est de distribuer les cellules entrantes aux bonnes sorties. Pour cela le noeud dispose d'une table de routage. Celle-ci associe au numéro de circuit virtuel la sortie à prendre par les cellules.

Problème : Le routage doit être rapide, on ne peut se permettre d'avoir un arbre binaire de recherche ou une table de hâchage. Par ligne, environ 360000 cellules arrivent toutes les secondes en conséquence le temps de cycle du commutateur est de 2.7 μs, en général, il y a 16 à 1024 lignes d'entrée. On aurait un temps de traitement par cellule de 700Ns (c'est très très court).

Donc : On voudrait utiliser [numéro de ligne, numéro de CV] comme index dans un tableau mais c'est impossible si numéro de CV est sur 28 bits. On route donc seulement sur les 12 premiers bits du numéro de CV. Ceci revient à grouper 2^{16} CV, en ce qu'on appele des conduits virtuels. Les CVs d'un même conduit virtuel prendront alors tous le même chemin.

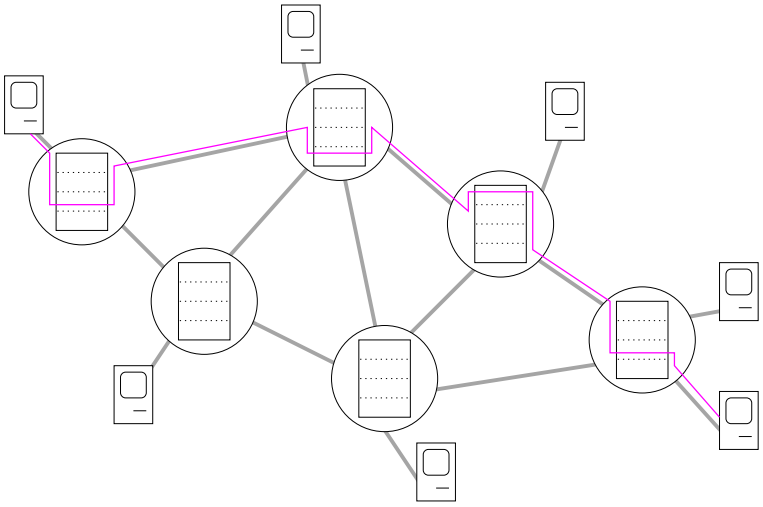

FIG. 4.9 – Le routage au long d'un circuit virtuel

4.7.5 Probleme du commutateur

Les cellules arrivent sur les entêtes. Grâce à la table d'index que l'on vient de voir, on peut imaginer que sur les cellules, il y a écrit un numéro de sortie (on peut, par exemple l'écrire dans le champs HEC, puisque de toute façon on doit le recalculer à l'émission). Il faut mettre les cellules vers les sorties.

S'il y a collision (plusieurs cellules pour une même sortie), on choisit une des cellules pour détruire les autres, où on met dans une file à capacité limitée. Nous présentons maintenant deux types de commutateurs.

4.7.5.1 Commutateur KNOCKOUT

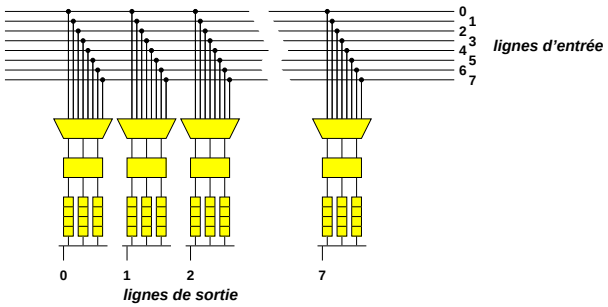

FIG. 4.10 – Le Commutateur KNOCKOUT

Supposons que nous ayons n entrées et sorties. Le commutateur présenté en figure 6 comporte un bus par ligne d'entrée, sur lequel sont connectés les dispositifs de sortie. Ils sont composés d'un filtre et concentrateur, qui récupère les cellules destinées à cette sortie et les concentre sur k files d'attente. Pour des raisons techniques il n'est pas possible de réaliser une unique file d'attente. Si à un moment donné plus de k cellules se présentent à une sortie, alors le concentrateur en choisit k, à la manière des tournois, d'où le nom de commutateur. Ensuite il y a un décaleur dont le rôle est de répartir les cellules uniformément sur les files d'attente. En faisant varier k, on peut régler le prix de ce commutateur versus le taux moyen de perte de cellules.

Le principal problème de ce commutateur est qu'il est de taille quadratique en n. Nous présentons maintenant un commutateur plus efficace.

4.7.5.2 Commutateur Batcher-Banyan

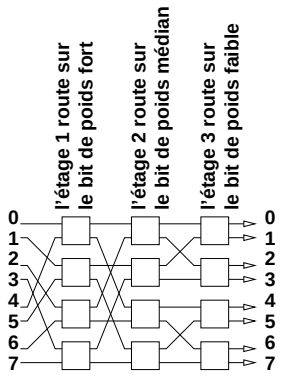

FIG. 4.11 – Le commutateur Banyan

Le nom Banyan vient d'un arbre indien à racines aériennes qui ressemble à la figure 4.11. L'élément de base est une boîte à deux entrées, deux sorties, qui route les cellules en fonction d'un bit particulier dans le numéro de sortie spécifié dans la cellule. La sortie haute prend les cellules dont ce bit est 0, la sortie basse ceux dont le bit est 1.

Si dans un élément deux cellules veulent prendre la même sortie une d'elles sera détruite. Par contre si les cellules se présentent dans l'ordre à l'entrée (dans l'ordre de leur numéro de sortie) alors il n'y aura aucune perte de cellule. D'où l'idée de faire précéder ce commutateur par un circuit qui trie les cellules.

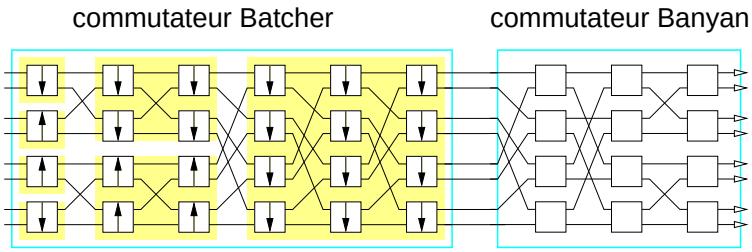

FIG. 4.12 – Le Commutateur Batcher-Banyan

Le premier circuit est composé d'éléments qui échangent les cellules qui se présentent à l'entrée quand ils ne sont pas dans l'ordre. Le tout est appelé le commutateur Batcher-Banyan et a seulement une taille $O(n \log^2 n)$.

4.7.6 Synchronisation

Les cellules ATM sont mises bout à bout lors de leur transmission et il se pose alors le problème à la réception de découper le flot de bits en flot de cellules, c'est-à-dire de retrouver le début des cellules. Dans HDLC par exemple ce problème était résolu à l'aide de fanions de séparation et un mécanisme de transparence par rapport à ces fanions. Mais ce n'est pas le cas avec les cellules ATM.

Le problème est résolu de la manière suivante. Les entêtes sont protégées par un code correcteur. Si beaucoup de cellules consécutives ont des entêtes incorrectes alors avec grande probabilité on est désynchronisé avec le flot de cellules. L'automate de la figure 4.13 décrit le mécanisme de synchronisation.

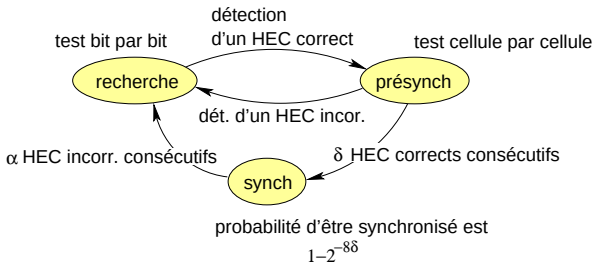

FIG. 4.13 – La synchronisation

4.8 Réferences

Ce chapitre était basé sur une note de cours de Julie Le Labourier. Le contenu du cours était basé sur le livre

– Réseaux de Andrew Tanenbaum, en français chez Dunod / Prentice Hall (1997)

Pour un apperçu sur les conflits entre industriels et comités de normes, lire

– "La Recherche", le numéro de février 2000.

Chapitre 5

La cryptographie

5.1 Introduction

La cryptographie est l'art de coder et de décoder les messages avec un cryptosystème. La cryptanalyse est l'art de décoder des messages interceptés. La cryptologie est l'union des deux domaines.

5.1.1 Quels sont les buts des cryptosystèmes ?

Confidentialité être sûr que personne d'autre ne puisse lire le message,

authentification être sûr de la personne à qui on parle,

non-répudation pouvoir prouver qu'un message a été envoyé,

Intégrité être sûr d'avoir reçu le message envoyé et pas un autre.

5.1.2 A quel niveau faut-il protéger les données ?

niveau physique Par exemple en entourant les câbles d'un tuyau rempli de gaz sous pression, muni d'un détecteur de baisse de pression. Dans la cryptographie quantique, les données sont codées sur des photons dont l'observation par un intrus peut être détectée.

niveau applicatif le secure socket layer (SSL) permet une transmission sûre. Kerberos est un système d'authentification au login qui ne révèle pas les mots de passe.

5.2 Schéma général (système à clé secrète)

Comme changer de méthode est parfois nécessaire mais coûteux, on préfère paramétrer la méthode avec une clé, qu'on peut changer facilement. On suppose alors que l'espion ne connaît pas la clé. Par contre il peut connaître la méthode utilisée.

5.2.1 Attaques possibles

Les attaques possibles de l'espion peuvent se classer comme :

attaque à texte chiffré seulement L'espion ne dispose que de quelques textes chiffrés et ne sait rien d'autre.

attaque à texte en clair connu L'espion dispose un couple texte clair/texte codé et doit en apprendre quelque chose sur la clé.

attaque à texte en clair choisi L'espion a la possibilité de coder des textes de son choix et de regarder le résultat codé.

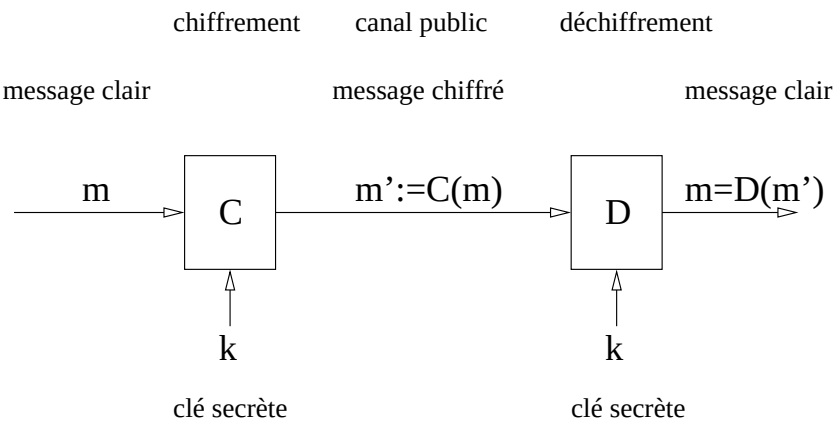

FIG. 5.1 – schéma d'un système à clé secrète

On pense parfois que seule la première attaque est réaliste. Mais souvent une partie du texte est connue, par exemple les sessions telnet commencent en général par "login :". Pour dire quelque chose sur la sécurité des schémas d'authentification, il faut distinguer en plus

l'espion passif qui ne peut qu'écouter les conversations échangées sur les canaux publics,

l'espion actif qui peut s'interposer dans une conversation, comme un passeur de seaux, et peut alors modifier les messages transmis.

5.3 Cryptographie traditionnelle

On apèlle cryptographie traditionnelle les systèmes anciens à clé secrète.

5.3.1 Code de Caesar

Il consiste à décaler chaque lettre de k lettres plus loin dans l'alphabet de manière circulaire.

Exemple pour $k = 3$, on notera le message en clair en minuscule et le message codé en majuscules : a→D, b→E, c→F, ..., z→C, et "attaque" se code en "DWWDTXH".

5.3.2 Code à substitution

Pour casser le code de Caesar, il suffit d'essayer 26 clés. Il n'est donc pas sûr. Une généralisation simple est le code à substitution, où la clé est une permutation sur les lettres.

Exemple de clé :

```
abcdefghijklmnopqrstuvxywz
LMNOEQSTPDUVXYWZABCJKFGHIR
```

le texte clair "attaque" se code en "LJJLAKE".

Est-ce que code est sûr ? Il y a 26! clés possibles. Si un ordinateur essaye 1 milliards de clés par seconde, il lui faudrait quand même 5 milliards d'années pour les essayer toutes. Par contre, une attaque utilisant les statistiques sur des textes français pour essayer les clés les plus plus probables en priorité rend ce système peu sûr.

5.3.3 Code à transposition

Dans ce code les lettres ne sont pas remplacées, mais changent de place. La clé est un mot ne comportant pas deux fois la même lettre. En fait ce mot ne sert qu'à coder un entier k (longueur de la clé) et un ordre sur les entier 1 à k (induit par l'ordre alphabétique sur les lettres). Ensuite le texte clair est noté ligne par ligne dans une grille à k colonnes. Le texte codé est lu colonne par colonne en suivant l'ordre donné par la clé.

Exemple : clé = BRIQUES
texte clair = demain a sept heures orly
transcription :

```
BRIQUES
demaina
septheu
resorly
```

La première colonne à lire est celle sous B, car c'est la plus petite lettre de la clé. La 2ème colonne à lire est celle sous E, car c'est la 2ème plus petite lettre de la clé. Et ainsi de suite.
texte codé = DSR NEL MPS ATO EEE AUY IHR

Ici le texte codé a été noté avec des espaces pour vous aider, ce qui ne sera bien sûr pas le cas en réalité, car on dévoilerait la longueur de la clé. Notez qu'il y a une petite difficulté lors du décodage si le texte codé n'a pas une longueur multiple de k.

Ce code peut de nouveau être attaqué en utilisant des statistiques sur les couples de lettres dans la langue française, d'abord pour deviner la longueur de la clé, ensuite pour deviner l'ordre sur les colonnes.

5.3.4 Combinaisons de codes

L'idée est de chiffrer plusieurs fois (en étages) en alternant un code à substitution et un code à transposition. C'est le principe de DES — *Data Encription Scheme* — introduit en 1977 et sujet de conflits entre l'Etat des Etats Unis et les industries sur le nombre d'étages. Il est utilisé pour les transferts banquaires ou dans le *secure socket layer*, mais n'est plus sûr depuis 1995. IDEA — *International Data Encription Algorithm* — est sensé le remplacer un jour.

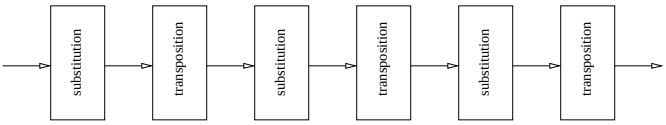

FIG. 5.2 – schéma très simplifié du DES

5.3.5 Bloc jetable

Un système à clé secrète extrêmement simple et sûr a été introduit par Gilbert Vernam. Soit un texte clair $c \in \{0,1\}^n$ de longueur n et une clé de même longueur $k \in \{0,1\}^n$. Alors le texte code est $c \oplus k \in \{0,1\}^n$ où $\oplus$ est le ou exclusif bit par bit. Pour déchiffrer il suffit de faire la même opération car $c \oplus k \oplus k = c \oplus 0 = c$. Ce système est parfaitement sûr, mais la clé est un peu longue et elle peut ne servir qu'une fois.

5.4 Cryptographie moderne

Le talon d'Achille des cryptosystèmes à clé secrète est que si un espion s'empare de cette clé, il n'y plus de sécurité. Il faut à la fois distribuer à tout le monde la même clé, et la garder secrète. Diffie et Hellman (Stanford) donnent en 1976 l'idée de chiffrement C (avec une clé publique) et un autre algorithme de déchiffrement D (avec une clé privée) qui doivent satisfaire

1. $D(C(M)) = M$ pour tout message M. (Pour beaucoup de cryptosystèmes nous avons aussi $C(D(M)) = M$, bien que ce ne soit pas nécessaire.)
2. Il est extrêmement difficile de déduire D à partir de C.
3. C ne peut pas être cassé (en temps raisonnable) par une attaque du type "texte en clair choisi".

La sécurité est basé sur l'hypothèse (largement acceptée) qu'il existe des fonctions faciles à calculer, mais difficiles à inverser.

Par exemple
387810*30187708 = _____ (facile)
338108371154320 = _____ * _____ (difficile)

Concrètement la fonction $x \mapsto x^a \bmod n$ pour certains n et a est candidate à cette hypothèse. Personne ne sait l'inverser efficacement, mais d'un autre côté personne n'a une preuve qu'il n'existe pas de méthode efficace pour l'inverser.

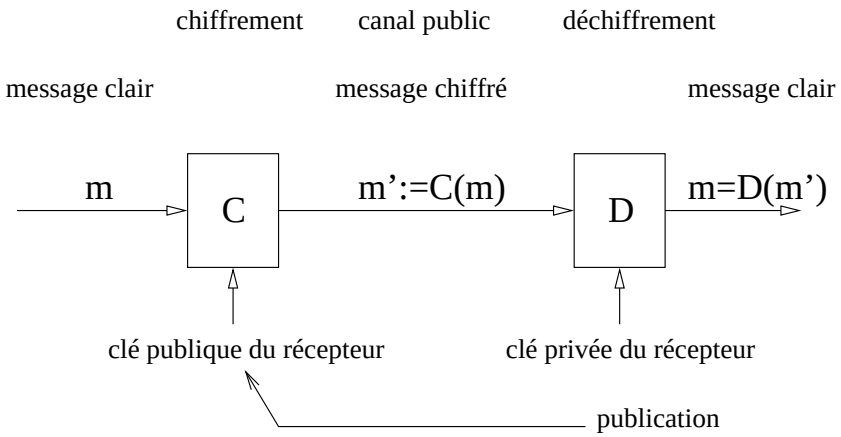

FIG. 5.3 – schéma d'un système à clé publique

Formellement, le destinataire du message doit avoir créé un couple de clés $\langle C, D\rangle$, publié C et bien caché D. L'émetteur envoie $C(M)$. Le destinataire y applique D pour récupérer le message M.

5.4.1 Exemple : RSA

Ce système a été inventé par Rivest, Shamir et Adleman. Le destinataire doit générer au hasard deux nombres premiers p, q. Il calcule $n = pq$ et $z = (p-1)(q-1)$. Puis il choisi un d premier avec z et un e tel que $ed \equiv 1 \pmod{z}$. La clé privée est alors le couple d, n et la clé publique e, n.

Un émetteur qui veut envoyer un message, codé par un entier $x \in \mathbb{Z}_n$ calcule puis envoie $y = x^e$. Tous les calculs sont fait dans $\mathbb{Z}_n$. Le récepteur calcule y^d ce qui est $x^{ed} = x$.

5.5 Authentification

Imaginons la situation : Je suis le processus de Jules et je veux détruire le fichier rapport.txt. Ne pas confondre :

authentification est-ce vraiment le processus de Jules ?

autorisation est-ce que Jules a le droit de supprimer ce fichier ?

5.5.1 Authentification basé sur une clé secrète partagée

Hypothèse : A et B sont les seuls à connaître une fonction C qui est difficile à inverser.

Ce schéma est basé sur un jeu de question/réponse, qui permet à chaque partie de se persuader que l'autre possède la clé sans toutefois la divulguer, comme c'est le cas lors d'une demande de mot de passe.

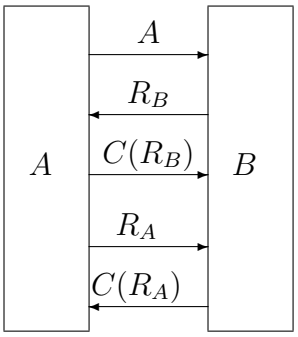

FIG. 5.4 – Authentification à l'aide d'une clé secrète partagée

A annonce à B qu'il veut ouvrir une session avec lui et prétend être A. Donc B sait qu'il doit utiliser la clé C, qu'il partage avec A. Ensuite B tire un nombre aléatoire R_B et demande à A de le coder avec la clé C. C'est ce que fait A, et B peut maintenant vérifier qu'il a bien reçu $C(R_B)$ et est maintenant sûr de parler à A. Mais A n'est pas encore sûr de parler à B, et va lui poser de manière symétrique le même type de question. A envoie un nombre aléatoire R_A et vérifie que B lui renvoie bien $C(R_A)$.

Les questions sont des nombres aléatoires pour éviter qu'une question ne soit posée plusieurs fois, et qu'un espion qui aurait enregistré la réponse la 1ère fois puisse se faire passer pour A.

5.5.2 Centre de distribution de clé

Le problème avec le schéma d'authentification précédent est que si 1000 personnes veulent pouvoir communiquer entre eux par paires de manière sûre, tout le monde doit garder presque 1000 clés et au total il faut à peu près 1 million de clés. L'idée du centre de distribution de clé est que chaque personne X ne garde qu'une clé K_X, celle qu'elle partage avec le centre et que le centre serve à la fois à authentifier les participants et à distribuer une clé qui servira juste pour la session.

A génère une clé K_S et l'envoie cryptée au centre. En même temps, elle ajoute en clair son nom, pour que le centre sache avec quelle clé déchiffrer et le nom de B pour indiquer avec qui elle veut ouvrir une session. Ensuite le centre extrait la clé et l'envoie chiffré à B en lui indiquant en même temps qu'elle vient de A.

5.5.3 Authentification à l'aide de la clé publique

Soient C_A, C_B les clés publiques de A et de B. A génère un nombre aléatoire R_A et l'envoie à B chiffré avec la clé publique de B. Ensuite B extrait R_A, génère à son tour un nombre aléatoire R_B et une clé de

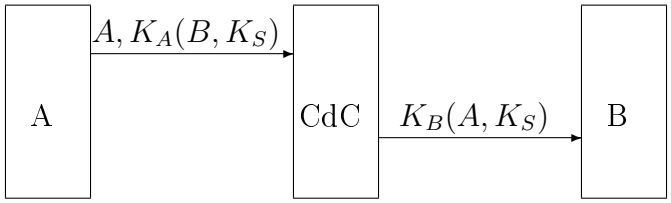

FIG. 5.5 – Authentification à l'aide d'un tiers de confiance

session K_S et les envoie chiffrés à A. Puis A extrait R_B et le renvoie chiffré avec la nouvelle clé de session K_S.

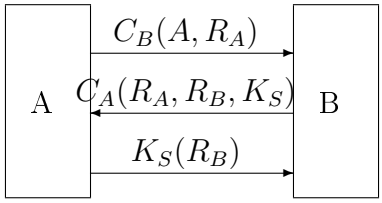

FIG. 5.6 – Authentification à l'aide de la clé publique

5.6 Signatures électroniques

Les signatures ont trois buts :

1. Le récepteur peut vérifier l'identité affichée par l'émetteur.
2. L'émetteur ne peut pas renier d'avoir écrit le document signé.
3. Le récepteur ne peut pas fabriquer lui-même un document signé.

Un système de signature électronique se résume à deux fonctions, une pour signer et une pour vérifier.

5.6.1 Résumé du message

Pour les signatures présentées dans les sections suivantes, leur taille est aussi grande que le message lui-même. Pour éviter ce problème, on peut calculer un résumé du message, qui sera plus court, puis signer celui-ci. Ce résumé devra être donné par une fonction de hachage satisfaisant les conditions suivantes :

1. La fonction de résumé f devra être facile à calculer.
2. Le résumé $f(M)$ ne permet pas de déterminer M (en temps raisonnable).
3. Personne ne peut engendrer deux message distincts M et M' tel que $f(M) = f(M')$ (en temps raisonnable).

La dernière condition impose qu'une signature fasse au moins 128 bits. La fonction la plus souvent utilisée est MD5.

5.6.2 Signature avec l'aide de clés publiques

Supposons que la clé publique C et la clé privée D satisfont pour tout message M : $C(D(M)) = M$ en plus de $D(C(M)) = M$ qu'ils satisfont par définition. Soient C_A, C_B et D_A, D_B les clés privées et publiques de A et de B.

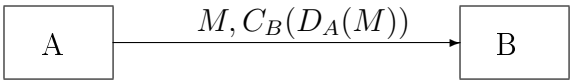

FIG. 5.7 – Signature avec l'aide de clés publiques

Alors A envoie le message M à B en même temps avec la signature $C_B(D_A(M))$. B est le seul à pouvoir vérifier cette signature. Il extrait $D_A(M))$ et vérifie à l'aide de la clé publique de A que $C_A(D_A(M)) = M$.

Bibliographie

[1] "Réseaux" de A. Tanenbaum. *Un livre complet, épais mais qui ne couvre pas les aspects pratiques.*

[2] "Computer Networking : A Top-Down Approach Featuring the Internet" de James F. Kurose et Keith W. Ross. *Une présentation originale et moderne, mais peut être plus adapté aux étudiants de 3ème cycle.*

[3] "Linux documentation project" http ://www.linux.fr/LDP/. *Contient de petits documents souvent bien écrit sur des aspects très pratique de l'administration réseau. Par contre "Administration réseau sous Linux" de Olaf Kirch est un peu dépassé par endroits.*

[4] "The Request for Comments (RFCs)", http ://abcdrfc.free.fr/. *Serie de documents techniques concernant Internet.*

[5] la page de ce cours http ://www.lri.fr/~durr/Iut/. *Contient un archive d'anciennes interrogations et du matériel necessaire pour les travaux pratiques.*