Réseaux – 2024-2025

révision 1.32

Marc Schaefer

évolution et extension du cours de Romain Voumard et d'Aïcha Rizzotti

28 août 2024

http://www.he-arc.ch/ingenierie

Préface

Les réseaux de télécommunication et bien sûr Internet sont aujourd'hui omniprésents. Il semble parfaitement logique que l'on puisse facilement échanger des données entre applications, terminaux mobiles et fixes, et interagir à tout moment.

Quand on considère les difficultés que l'informatique n'a toujours pas pu résoudre (compatibilité entre applications, viabilité des formats de données, coût de mise à niveau et de portage d'application – et ceci même au sein d'un même éditeur de logiciel), le fait que les réseaux modernes communiquent entre eux de manière fiable et **interopérable** ne peut pas venir de soi.

Cela n'est bien sûr possible que grâce à des standards universels adéquats, reconnus comme tels, et implémentés sur tous les équipements (p.ex. la pile de protocoles IP) et à un modèle clair d'interaction et de découpage des fonctionnalités en couche, fruit de dizaines d'années de recherche et d'expérimentation.

Aujourd'hui, l'informatique et les télécommunications ont fusionné : les réseaux informatiques transportent des informations multimédia, y compris des communications téléphoniques (voix-sur-IP), et les réseaux de télécommunications acheminent les données informatiques à travers le globe, que ce soit sur Internet ou dans des infrastructures privées. Les opérateurs classiques de télécommunication sont en pleine mue pour s'adapter à la nouvelle donne des services réseau de prochaine génération (**Next Generation Networks**).

Le contenu du cours Réseaux traite notamment

- des modèle en couches qui peuvent être utilisé pour décrire toute architecture de réseaux (télé-)informatiques ou de télécommunications (le modèle OSI)
- des protocoles utilisés aujourd'hui dans la plupart des couches concernées
- de la configuration de réseaux classiques (bases théoriques et pratiques solides)
- des outils et méthodologies pour décrire et trouver des problèmes au sein du réseau
- de la conception et la réalisation d'applications réseau

Ce document se veut un ouvrage de référence qui vous accompagnera lors du cours.

Sommaire

Chapitre 0

Introduction aux réseaux et à Internet

Sommaire

Le but de ce chapitre est de décrire les modèles et standards qui permettent à une **application réseau** de communiquer de manière **interopérable** entre ses **instances** réparties sur des réseaux **hétérogènes** (de constructeur, de matériel et de versions différents).

0.1 Les réseaux informatiques

0.1.1 Introduction

0.1.1.1 Fondements

Les réseaux informatiques sont, très généralement, des ensemble d'équipements (par exemple d'ordinateurs) géographiquement plus ou moins distants les uns des autres, interconnectés par des systèmes de **télécommunication** généralement permanents.

A l'origine, les systèmes informatiques étaient basées sur des ordinateurs centraux reliant à basse vitesse (quelques kilobits [1] par seconde) des périphériques éloignés de quelques dizaines de mètres à l'aide câbles coaxiaux ou multifilaires en cuivre. La variété des systèmes de câblage retenus par les différents constructeurs a conduit à des **topologies point à point** très entrelacées, incompatibles et peu souples.

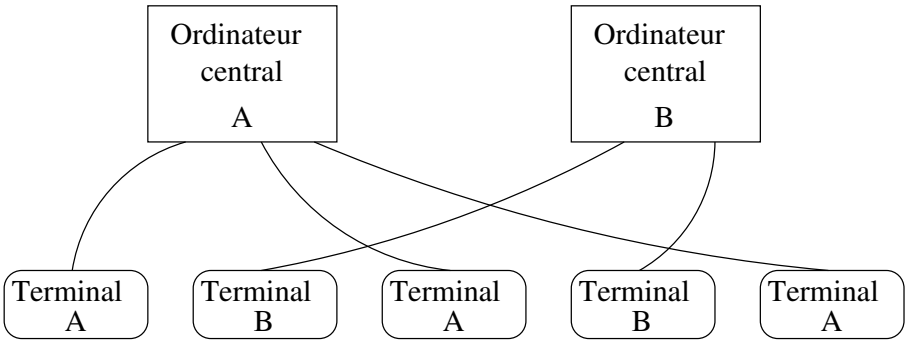


FIGURE 1 – Exemple de configuration non réseau

La configuration ci-dessus ne permet une communication directe qu'entre un ordinateur central et un terminal. Ni les terminaux ni les ordinateurs ne peuvent communiquer directement entre eux. Avec l'avènement de systèmes informatiques plus complexes et de stations de travail plus puissantes, il est devenu nécessaire que chaque utilisateur et chaque ordinateur puissent communiquer directement avec les autres. L'utilisation de lignes point à point n'est évidemment pas adaptée à cette situation. C'est pourquoi de véritables **réseaux** sont apparus.

0.1.1.2 Eléments d'un réseau

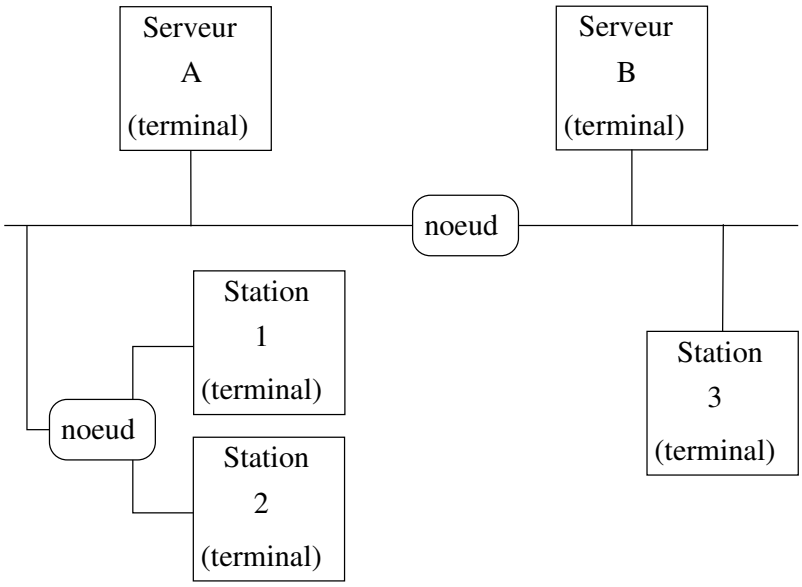


FIGURE 2 – Configuration réseau

Un réseau permet à chaque utilisateur de communiquer avec tous les autres sans qu'une énorme *salade* de câbles n'en résulte, notamment en raison de la **liaison point à multipoint** (**PTMP**).

1. en téléinformatique, on utilise souvent le bit par seconde plutôt que l'octet ou le byte (== 8 bits) par seconde pour mesurer les débits ; le préfixe kilo est celui du système international d'unité (SI) et vaut donc 1000 – contrairement au stockage informatique qui utilise souvent des puissances de 2 et donc des multiples de 1024.

Les principaux éléments d'un réseau (voir figure 2 en page 2 et figure 7 en page 15) sont [2] :

— les feuilles ou **terminaux** (littéralement ceux qui sont en bout de ligne : serveur [3], PC, station de travail, imprimante, capteur...)
— les **nœuds** qui ont comme tâche de faire suivre les données entrantes sur la bonne ligne sortante (en couche 1 : **concentrateur** ou **hub**, en couche 2 : **commutateur**, **bridge** ou **switch**, en couche 3 : **routeur**)
— les **lignes** qui relient entre eux les composants du réseau ci-dessus

Ne pas confondre :

— les définitions ci-dessus sont liées à la théorie des graphes : en couche 3 IP, on utilise souvent la notion de **node** ou **nœud** pour désigner *tous* les équipements de couche 3 ayant une adresse IP (en particulier les routeurs : noeuds intermédiaires *et* terminaux : noeuds-feuilles)
— la notion de **terminal** désigne en informatique un dispositif d'interface homme machine : p.ex. clavier et écran et par extension les terminaux virtuels et consoles dans lesquels un shell comme `bash` s'exécute sous UNIX.

0.1.1.3 Hétérogénéité et interopérabilité

Dans un tel réseau, la tentation de connecter des appareils de différents constructeurs (**hétérogènes**) est plus grande, en particulier pour pouvoir rentabiliser l'infrastructure commune (le réseau) pour relier toutes sortes d'appareils et pour transmettre toutes sortes de d'informations (données informatiques mais aussi la voix ou de la vidéo). Pour réaliser ces exigences il faut *normaliser* l'accès au réseau, pour que les nœuds ne doivent pas connaître trop de protocoles différents, et pour que des appareils de différents constructeurs se comprennent.

Les normes doivent assurer l'**interopérabilité** des composants du réseau. L'intéropérabilité est *la capacité que possède un produit ou système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre* [4]. L'interopérabilité n'impose toutefois aucune implémentation spécifique et ne concerne que les interfaces du système.

Il y a une très grande différence entre l'**interopérabilité** (par normes et standards) et la **compatibilité** (par convertisseurs) :

— temporellement, l'interopérabilité est conçue (et *voulue*) au moment où le **protocole** (le *langage commun et unique* sur le réseau) est défini, alors que la compatibilité est une mesure après coup, pour étendre une part de marché
— l'interopérabilité est moins complexe : en effet, la compatibilité exige une implémentation de couche intermédiaire de conversion par paire de partenaires : s'il y a N partenaires, il faut $\frac{N(N-1)}{2}$ couches de compatibilité [5], alors que l'on pourrait parler *un seul langage commun* sur le réseau
— l'interopérabilité peut faire l'objet de tests reconnus
— les convertisseurs perdent souvent des fonctionnalités

2. on peut bien sûr y ajouter les notions d'utilisateur et d'**agent** (logiciel qui va agir pour l'utilisateur)
3. host, mainframe
4. http://definition-interoperabilite.info/
5. il s'agit d'une complexité de l'ordre du carré, ce qui est en général à éviter dans tout système.

0.1.1.4 Besoins

Apparus durant les dernières décennies, ces réseaux informatiques font partie d'une évolution majeure dans le monde des ordinateurs. C'est l'avènement des stations de travail complexes et de la répartition du traitement de l'information (réseau de calcul, réseau de données) qui est à la base du développement des réseaux informatiques.

Dans un réseau, contrairement à un système *non réseau* qui travaille en point à point, les lignes peuvent être mieux utilisées (**mutualisation** des ressources et donc des coûts). Cette mutualisation crée toutefois de nouveaux problèmes à résoudre, notamment la nécessité de gérer les accès concurrents.

De plus, la transmission de données informatiques est par nature numérique et irrégulière en durée et en volume. La **commutation de lignes** (voir section 0.1.3 en page 5), originellement appliquée à la téléphonie, n'est plus adéquate : on passe à une **commutation de paquets**, où les données sont découpées en petites unités de taille variable[6], les **paquets**, **multiplexés** les uns derrière les autres (voir section 1.2.7 en page 27). C'est la meilleure solution pour mutualiser le réseau et optimiser le temps de transfert et donc le délai minimum.

0.1.1.5 Rôle du réseau informatique

On peut définir un réseau informatique comme étant l'ensemble des outils qui permettent à plusieurs systèmes intelligents de communiquer à travers l'espace géographique qui les sépare. Cette définition porte donc sur les domaines du transport physique de l'information (câblage), de la méthode d'accès à l'information et de la représentation de l'information (unité d'information ou paquet), du détail de l'adressage des paquets (protocoles), des formats de données et des applications de communication proprement dites (transfert de fichiers, partage de ressources, web, etc) : ces domaines sont traités en couches séparées (voir section 0.2.3 en page 8).

0.1.2 Types de réseaux

On peut distinguer plusieurs types de réseau par leur envergure et leur rôle :

- le **réseau personnel** ou **PAN** (*Personal Area Network*), qui peut être implémenté en partie avec un LAN (avec ou sans-fil) ou d'autres protocoles reliant des équipements personnels : ordinateur, tablette, équipement portable de mesure de paramètres de la personne (**wearable sensors**[7])...
- le réseau local **LAN** (*Local Area Network*, ou **RLE** : *Réseau local d'entreprise*) dont le but principal est d'assurer le transfert de données à haute vitesse (de quelques centaines de mégabits à quelques gigabits par seconde) entre systèmes distants de quelques centaines de mètres (PC, serveurs, imprimantes...)
- le réseau urbain **MAN** (*Metropolitan Area Network*) qui vise le même but à l'échelle d'une ville (quelques dizaines de kilomètres) : par exemple le réseau d'un opérateur télécom ou d'un **FAI**
- le réseau **WAN** (*Wide Area Network*) qui fait appel aux systèmes de télécommunications mutualisés (satellites, fibre optiques terrestres et sous-marines...) jusqu'au globe entier

mais aussi :

- le réseau fortement couplé (**bus local**) qui assure l'interconnexion à très haute vitesse (quelques centaines ou milliers de mégabits par seconde) mais à faible distance des

6. de taille bornée au *Maximum Transmission Unit*, le **MTU**
7. on peut même alors parler de **BAN**, *Body Area Network*

composants internes de l'ordinateur (CPU, RAM...) : citons par exemple le bus PCI-Express

— le **bus de terrain**, qui relie en filaire ou sans-fil des équipements de mesure (smart-meters) ou de contrôle à l'intérieur d'une voiture (bus CAN, avec des services **temps réel**), d'une ligne de production (Modbus, Profibus) ou d'un bâtiment (Z-Wave), voire d'une région (**LoRaWAN**), avec une intégration possible aux réseaux classiques en particuliers lorsque des applications informatiques y sont interfacées (aspects de haut niveau de gestion, **Industrie 4.0**) – certains types sont des **PAN**

0.1.3 La mutation des réseaux téléphoniques classiques

Le réseau téléphonique public a joué un rôle particulier dans l'évolution des réseaux de communication. Ce réseau est le premier[8] à s'être répandu sur l'ensemble de la planète. Grâce à une bonne normalisation, l'interopérabilité entre les opérateurs est excellente. Ce réseau a été évidemment conçu en premier lieu pour le transport de la voix. Mais il était évidemment tentant de l'utiliser aussi pour le transport des données à cause de son ubiquité[9].

Il s'agissait d'un réseau public qui permettait le transport de la voix sur des distances à l'échelle planétaire par une technique appelée **commutation de circuit**. Chaque central intermédiaire devait mettre à disposition une ligne vers le destinataire. On établissait, à l'aide de commutateurs d'abord mécaniques, puis électroniques, une liaison physique qui reliait l'appelant à l'appelé à travers diverses lignes de cuivre permanentes, pour la durée de la communication. La tarification des communications était basée logiquement sur la distance et sur la durée. La transmission de données numériques y était possible grâce la conversion préalable de données digitales en données analogiques compatibles, à l'aide d'un équipement appelé **modem** (modulateur-démodulateur), qui convenait bien à une transmission ponctuelle de petits volumes de données. L'exemple du télécopieur (téléfax) qui n'était rien d'autre qu'un digitaliseur d'images (scanner) doublé d'un modem illustre parfaitement ce domaine d'application. Par contre, le débit offert n'était pas suffisant pour les gros volumes de données et la tarification sur la durée était trop pénalisante.

Pour cette raison, un réseau public de **commutation de paquets** à été développé (**X.25**). Un débit supérieur (2 à 3 fois) et une tarification basée sur la distance et le volume ont permis son développement rapide dans la plupart des pays industrialisés.

Ces deux types de réseau sont depuis quelques années supplantés[10] par ceux basé sur les protocoles d'Internet (TCP/IP). On a aujourd'hui, à côté des réseaux classiques de télécommunication, un réseau mondial, Internet, qui utilise l'infrastructure de télécommunication classique ou, de plus en plus, ses propres infrastructures, pour échanger ses paquets. Le réseau téléphonique sert encore de porte d'entrée, que cela soit uniquement sur le dernier kilomètre, p.ex. avec les technologies **xDSL** comme l'**ADSL** ou le **VDSL**, ou dans de rare cas, à travers de modems analogiques classiques.

A l'intérieur des entreprises, on trouve généralement un réseau téléphonique local de type commutateur privé P[A]BX (*Private [Automatic] Branch Exchange*), de type analogique, numérique (ISDN/RNIS) ou en voix-sur-IP, connecté à un réseau public de télécommunications externe.

Depuis 2018 en Suisse, SWISSCOM ne supporte que la **voix-sur-IP** : les équipements ISDN et de téléphonique analogique (**POTS**, *Plain Old Telephone System*) ont été tout bonnement

8. à l'exception du réseau télégraphique mondial, aujourd'hui obsolète
9. il est disponible partout
10. quasi totalement pour X.25 sauf p.ex. pour des applications de validation de carte de crédit ou des pays sans infrastructure Internet.

supprimés.

0.1.4 Avenir des réseaux : vers la fusion tout IP

La téléphonie mobile complète les possibilités du réseau téléphonique. Dans ce domaine on a observé une évolution similaire à celle du réseau fixe : dans un premier temps le transport de la voix (GSM) puis une évolution vers le transport des données (HSCSD, GPRS puis EDGE, UMTS et enfin 4G/LTE – *Long Term Evolution*, puis 5G), pour aboutir à une infrastructure **all-IP** (tout en IP, *Internet Protocol*).

Avec ISDN et les modems, on utilisait l'infrastructure faite pour le transport de la voix pour transférer des données. Désormais, on fait usage de réseaux conçus au départ pour le transport de données pour transporter la voix (**voix-sur-IP**, **VoIP**) : ce réseau **all-IP**, utilisant uniquement des protocoles paquets basés IP (Internet Protocol), évolution des technologies actuelles de télécommunications, nécessite de mettre en place de la qualité de service (**QoS**) répondant aux besoins des différentes applications (informatiques, voix-sur-IP...), par exemple à travers les réseaux de prochaine génération des opérateurs télécoms et FAIs **NGN-IMS**. Des passerelles permettront d'exploiter les anciennes technologies en parallèle (voir section 3.3.4 en page 78).

0.2 Les normes et standards

0.2.1 Organismes de normalisation

De nombreux acteurs des mondes des télécommunications, de l'Internet ou des constructeurs définissent des normes, qui ont offert une stabilité, une ouverture et ont été à la base du développement très important des services proposés. Les sections suivantes listent les plus importants organismes de normalisation.

Il est à noter que ces organismes ne sont pas strictement neutres : ils reflètent l'écosystème dans lequel ils ont été créés. Par exemple, les normes de télécommunications reflètent un monde dans lequel les opérateurs télécom fixent les règles ; et les normes du monde Internet prennent le point de vue des besoins téléinformatiques et des grands acteurs de l'Internet.

0.2.1.1 Internationaux

ISO	International Organisation for Standardization
CEI	Commission Electrotechnique Internationale
UIT	Union Internationale des Télécommunications (anglais ITU, ex-CCITT)
ICANN	Internet Corporation for Assigned Names and Numbers
IFRB	International Frequency Registration Board
IETF	Internet Engineering Task Force
W3C	World Wide Web Consortium
OASIS	Openstandards for the information society consortium

L'IETF, un groupe informel, sous l'égide de l'ONG Internet Society, définit les normes Internet via le processus ouvert des **RFC** (*Request For Comments*), dont certains sont des standards reconnus (**STD-xxxx**) ou des recommandations de meilleures pratiques (**BCP**, *Best Current Practice*). L'ICANN, société de droit californien à but non lucratif, est responsable des espaces

de noms d'Internet (attribution des adresses, des identificateurs de protocoles, des noms de domaine de premier niveau, etc). Quant à l'ITU-T, elle définit de nombreuses normes touchant l'ensemble d'un système de télécommunication. Enfin, citons le W3C (HTML5) et l'OASIS qui a par exemple défini l'Open Document Format – aussi un standard ISO (ISO/IEC 26300 : 2006).

0.2.1.2 Aux Etats-Unis d'Amérique (USA)

ANSI	American National Standards Institute
IEEE	Institute of Electrical and Electronics Engineers
EIA	Electronic Industries Association (Recommended Standards)
FCC	Federal Communication Commission
NIST	National Institute for Standards and Technology

De nombreuses normes ont été définies par l'IEEE, citons par exemple l'IEEE-1394 ou Firewire, ou l'ensemble des normes liées aux réseaux Ethernet et assimilés (802.x).

0.2.1.3 En Europe

CEN	Comité Européen de Normalisation
CENELEC	Comité Européen de Normalisation ELECtrotechnique
CEPT	Conférence Européenne des Postes et des Télécommunications
ECMA	European Computer Manufacturer Association
ETSI	European Telecommunications Standard Institute
ECITC	European Committee for Information technology Testing and Certification
EWOS	European Workshop for Open Systems
DIN	Deutsches Institut für Normung
AFNOR	Association Française de NORmalisation

Citons par exemple les normes ECMA-262 et ECMA-402 qui définissent le langage ECMAScript, à la base du JavaScript, langage côté client associé à l'écosystème HTML5 du web.

0.2.2 Architectures d'échanges

0.2.2.1 Introduction

Les interactions entre applications sur un réseau peuvent se faire selon une ou plusieurs architectures : les plus communes sont l'architecture client/serveur et pair-à-pair. Elles peuvent bien sûr être combinées, aussi de manière hiérarchique.

0.2.2.2 Client/serveur

L'architecture client/serveur est le modèle classique d'Internet : un client pose des questions à un serveur. Ce modèle est asymétrique, et plus ou moins de complexité peut résider du côté client ou du côté serveur, selon qu'on implémente un **client léger**, **riche**, ou **lourd**. Le serveur quant à lui peut être monolithique ou découpé en N partenaires (**N-tier**), en particulier si l'on déploie des **microservices**.

Les échanges peuvent être strictement initiés par le client de manière synchrone (**PULL**), avec **scrutation** pour déterminer si le serveur a quelque chose à dire, ou, mieux, des notifications – auxquelles le client peut s'être abonné – peuvent être envoyées par le serveur de manière asynchrone (**PUSH**).

0.2.2.3 Pair-à-pair (P2P)

L'architecture P2P (*peer-to-peer*) considère des applications sans point central, ou avec un point central (serveur) utilisé uniquement pour la localisation de services (annuaire). Les applications communiquent directement entre elles au besoin. Ce modèle est rendu nécessaire dans les applications fortement interactives ou pour lesquelles une forte performance ou résistance aux pannes est souhaitée, mais peut être complété par un mode de repli en client/serveur centralisé en cas de besoin (par exemple présence de pare-feu). En première approche, on peut l'implémenter avec des clients qui parfois se comportent en serveur et grâce à des annuaires où leurs services (**endpoint**) s'enregistrent. Ne pas confondre avec la **topologie point à point** (**PTP**, voir section 1.3.1 en page 27).

0.2.3 Modèle de référence OSI

0.2.3.1 Principes

L'organisation **ISO** a proposé en 1983 un modèle de référence en sept couches (voir figure 3 en page 9) pour tenter de normaliser la connexion entre **systèmes ouverts** pour atteindre l'**interopérabilité** (voir section 0.1.1.3 en page 3).

Sa dénomination anglaise *Open Systems Interconnection* est à l'origine de l'abréviation courante OSI. Ce modèle est un exemple de la méthode *subdiviser pour régner* appliquée à la transmission depuis le niveau physique jusqu'au niveau le plus haut. Les principes ayant conduit aux sept couches sont les suivants :

- une couche doit être créée lorsqu'un nouveau niveau d'abstraction est nécessaire.
- chaque couche exerce une fonction bien définie.
- les fonctions de chaque couche doivent être choisies en pensant à la définition de protocoles normalisés internationaux.
- le choix des frontières entre couches doit minimiser le flux d'information aux interfaces.
- le nombre de couches doit être suffisamment grand pour éviter la cohabitation dans une même couche de fonctions très différentes et suffisamment petit pour éviter que l'architecture ne devienne difficile à maîtriser.

Le modèle à 7 couches OSI est le modèle standard utilisé en télécommunication et téléinformatique pour décrire des systèmes interconnectés communiquants. Ce modèle a cependant été conçu après de nombreux développements, notamment TCP/IP. C'est pour cela que la correspondance n'est pas toujours parfaite dans ce contexte : un modèle reste un *idéal*.

Attention : de nombreux protocoles Internet considérés de couche 7 ont *transport protocol* dans leur nom (p.ex. SMTP, Simple Mail Transport Protocol), ce qui peut prêter à confusion (voir section 0.2.4 en page 18).

0.2.3.2 Les couches

0.2.3.2.1 Couche physique La couche physique s'occupe de la transmission des bits de façon brute sur un circuit de communication. Sa conception doit être telle que l'on soit rai-

couche	**nom**	**rôle**	**protocoles ; équipements ou services**
7	application	supporte le protocole applicatif utilisé et gère la communication entre applications	SMTP, HTTP, DNS, NTP ; **proxy** et **gateway**
6	présentation	uniformise ou convertit la représentation de l'information : encodage, chiffrement	**encodages** et **charset**, types **MIME**, encodages de transfert **base64**, format des données (JSON, XML...), compression (gzip) et chiffrement SSL/TLS
5	session	organise les sessions de communication entre applications : ouverture, fermeture, reprises de contextes, authentification	session, cookie, token
4	transport	relie les couches supérieures de manière transparente à la structure du réseau (première liaison logique de bout en bout : d'application à application) : transmission fiable (ou non) des données à travers un réseau	TCP, UDP
3	réseau	gère l'adressage et le routage dans un réseau, établit la liaison (l'échange) entre systèmes finaux (machine à machine)	IP (Internet Protocol) ; **routeur**
2	liaison	assurer la transmission de données au sein d'une liaison point à point ou multipoint, gère les erreurs, délimite les trames	Ethernet, HDLC ; **switch**
1	physique	transmission physique de l'information (interface électrique et mécanique, conversion des données en signaux adaptés au média)	interface, câblage, signaux ; **hub**

FIGURE 3 – Modèle OSI à 7 couches

sonnablement sûr que les valeurs des bits ne sont pas mal interprétées (en valeur et au bon moment). Les questions portent donc sur le nombre de volts (ou d'autres grandeurs physiques, éventuellement combinées) à atteindre pour représenter un bit à 1 et à 0, la durée d'un bit en microsecondes et sa synchronisation, la possibilité de transmission dans les deux directions simultanément [11], l'initialisation et la libération de la connexion, le nombre de broches du connecteur et l'usage de chacune. D'autres considérations (p.ex. tension moyenne électrique, présence ou non de l'horloge dans le signal, etc) sont également à prendre en compte dans la modulation et le mode de transmission choisis (voir section 1.4.3 en page 31).

Le protocole de cette couche s'applique à un média donné (une ligne, un canal de transmission ; répéteurs/hubs inclus). Chaque ligne d'un réseau couche 3 peut utiliser un protocole différent en couche 2 et 1 (notion de portée, voir section 0.2.3.6 en page 16).

11. mode **full-duplex**, opposé aux modes **half-duplex** (à l'alternat) et **simplex** (une seule direction), voir section 1.2.6 en page 26

0.2.3.2.2 Couche liaison de données Le tâche principale de la couche liaison de données est de prendre un moyen de transmission *brut* proposé par la couche physique et d'y ajouter des fonctionnalités utiles pour la couche réseau : il s'agit d'organiser les données en trames [12] (paquets) envoyées en séquence, qui sont adressées et complétées de détection, voire de correction d'erreur par **redondance** ou même de retransmission. Une autre fonction de cette couche est d'empêcher un émetteur rapide d'inonder de données un récepteur lent : un mécanisme de régulation de trafic peut être employé pour porter à la connaissance de l'émetteur la capacité instantanée de traitement (**contrôle de flux**).

Le protocole de cette couche s'applique à une ligne ou un canal de transmission : la problématique de l'accès multiple au canal doit également être gérée. Des nœuds intermédiaires (commutateurs/switches) peuvent être utilisés. Chaque ligne d'un réseau couche 3 peut utiliser un protocole différent en couche 2 et 1 (notion de portée, voir section 0.2.3.6 en page 16).

0.2.3.2.3 Couche réseau La couche réseau permet aux informations, transportées sous forme de paquet, de traverser un réseau constitué de couches 2 et de nœuds (routeurs). La façon dont ces paquets sont acheminés de la source au destinataire constitue un élément clé de la conception : le **routage** peut être fondé sur des tables de routes statiques, ou mieux sur des tables dynamiques (qui peuvent être modifiées en cas de changement de topologie, ou, mieux pour refléter la charge du réseau).

Les lignes et les routeurs peuvent devenir surchargés : le contrôle de **congestion** peut aussi appartenir à la couche réseau, ainsi que le concept plus général de **traffic engineering** : le réseau peut contrôler le respect d'un contrat de service (**SLA** – *Service Level Agreement*). Les règles appliquées à un flux donné peuvent dépendre de la qualité de service choisie (**QoS** : critères : débit, délai et variation du délai ou *jitter*, taux de perte ...). Dans certains cas, le volume peut être facturé.

Le protocole de cette couche est le premier qui peut être de **portée** universelle (voir section 0.2.3.6 en page 16) : il relie deux systèmes finaux (terminaux) du réseau (par exemple 2 PC), à travers un certain nombre de couches liaison (et donc physique) différentes, unifiées au sein du *même* réseau couche 3.

0.2.3.2.4 Couche transport La fonction de base de la couche transport est d'accepter des données de la couche session, de les découper en plus petites unités si nécessaire, et de s'assurer que tous les morceaux arrivent correctement à leur destinataire dans le bon ordre (**protocole fiable**, avec retransmission [13] lorsque l'**acquittement** (confirmation) par le récepteur n'arrive pas) et sans le surcharger (**contrôle de flux**). Normalement, la couche transport crée un flux ou une connexion réseau par flux ou connexion de transport requise par la couche session. Cependant, si le flux ou la connexion de transport requiert un débit rapide, la couche transport pourrait créer de multiples connexions ou flux réseau, sur lesquelles elle répartirait les données pour atteindre le débit souhaité (**load balancing**). A l'inverse, dans le cas où la création et le maintien d'une connexion sont coûteuses, la couche transport peut *multiplexer* plusieurs flux ou connexions de transport sur le même flux ou la même connexion réseau pour réduire le coût : c'est le cas le plus courant.

La couche transport est une authentique couche de *bout en bout*, de l'émetteur au destinataire. Autrement dit, un programme de la machine source véhicule une conversation avec un

12. la couche physique ne livre ni transmet qu'un flot de bits sans en connaître la signification ni la structure, c'est à la couche liaison de créer et reconnaître les frontières des trames, en accolant par exemple des groupes spécifiques de bits au début et à la fin de chaque trame (**fanions**), voir section 2.2 en page 38.

13. tout en gérant la problématique des **duplicats** causés par la perte d'un acquittement, grâce aux numéros de séquence

programme similaire sur la machine cible, à l'aide de messages d'en-tête et de contrôle : on dit que les instances de couche 4 communiquent.

0.2.3.2.5 Couche session La couche session permet à des applications sur différentes machines d'établir des sessions entre elles. Une session peut permettre à un utilisateur d'accéder à un système distant en mode graphique (X11, RDP) ou texte (SSH), ou de transférer un fichier entre deux machines. Un des services de la couche session concerne la gestion du dialogue. Les sessions peuvent autoriser le mode bi- ou unidirectionnel du trafic, mais il est parfois essentiel de pouvoir garantir que les deux côtés ne lancent pas la même opération en même temps. Un autre service de session est la resynchronisation en cas d'interruption ou la persistance d'une unité de session à travers plusieurs connexions ou flux de couche 4, même après le redémarrage d'un programme (exemple : session HTTP, poursuite d'un achat). Enfin, cette couche peut également modéliser du contexte persistant, par exemple de l'authentification, sous forme de **cookie** ou de **token**.

0.2.3.2.6 Couche présentation La couche présentation effectue des fonctions suffisamment courantes pour ne pas être laissées à la charge de chaque application. Elle s'intéresse particulièrement à la **syntaxe** (la forme) de l'information transmise. Un exemple typique d'un service de présentation est l'encodage des données dans une norme agréée : les machines et applications pouvant représenter les données selon des codes différents (chaînes en ASCII, ISO-Latin-1, UTF-8 ... ; valeurs en **little endian** ou **big endian**, etc).

La structure des données à échanger peut être définie de façon **abstraite**, afin de permettre la communication entre des ordinateurs ayant différentes représentations internes. La couche présentation est aussi concernée par d'autres aspects de la représentation de l'information. Ainsi, la compression des données peut être utilisée pour réduire le volume transmis, et le chiffrement et la signature numérique (voir section 6.6 en page 104) sont fréquemment nécessaires pour la confidentialité et l'authentification.

0.2.3.2.7 Couche application La couche application comporte de nombreux protocoles fréquemment utilisés. Par exemple, beaucoup de protocoles Internet classiques sont ligne-à-ligne, en format ASCII (texte simple non accentué), basé sur des connexion TCP et une notion de terminal virtuel de réseau (**NVT**).

Un de ces protocoles est **HTTP**, qui est à la base de nombreux protocoles de plus haut niveau : citons le classique transfert de documents web, l'échange de fichier **WebDAV** utilisé notamment pour les calendriers en-ligne ou le partage de fichiers, ou encore la philosophie **REST** qui permet de construire des applications orientées services (APIs **Web Services**).

On considère souvent que ces protocoles de plus haut niveau ainsi que l'application proprement dite font partie de cette couche.

0.2.3.3 Interactions entre les couches

Chaque couche offre des services qui sont spécifiés dans le modèle OSI. Elle exécute ces services à la demande de la couche située directement au-dessus d'elle. Elle utilise les services de la couche située au-dessous d'elle au cours de la réalisation de ces services.

Chaque couche est découpée en (au moins) deux parties : chaque partenaire contient de la logique pour exécuter les services de la couche. La logique située chez un partenaire est appelée **instance** de la couche. On a toujours au moins deux instances dans une couche. Les instances

collaborent pour la réalisation des services de la couche. Elles communiquent entre elles à l'aide d'un protocole. Elles utilisent les services de la couche directement inférieure pour communiquer et échanger des messages (**PDU**, *Protocol Data Unit*). La répartition des tâches entre les instances et le protocole qu'elles utilisent pour communiquer entre elles, sont opaques (invisibles) pour la couche supérieure. Celle-ci ne voit qu'un fournisseur de services qu'elle peut utiliser.

Lors du transfert d'un message d'une application à une autre, l'application émettrice génère le message qu'elle souhaite envoyer et elle le passe à la couche de présentation afin que celle-ci d'une part exécute ses services sur ce message et, d'autre part, en assure le transport [14] jusqu'à l'instance partenaire de la couche d'application. La couche de présentation va exécuter ses services pour ce message et le passer à la couche de session pour qu'elle aussi exécute ses services et prépare le transport du message jusqu'à l'instance distante de la couche de présentation. Ce schéma se répète jusqu'à la couche physique qui transmet enfin *réellement* les données à l'instance partenaire de couche physique.

Afin de pouvoir réaliser les services de la couche pour le message transmis, les instances doivent ajouter au message des informations de contrôle (adresses par exemple) destinées à leur instance partenaire. Ces informations de contrôle de la couche sont insérées devant (et éventuellement après en couche 2, pour la redondance structurée permettant la détection, voire la correction d'erreur) le message reçu de la couche supérieure. Le tout forme une unité de message du protocole considéré (**PDU**, *Packet Data Unit*) :

Les instances du côté du destinataire analysent et traitent les informations de contrôle et les suppriment du message avant de passer celui-ci à la couche supérieure. La réalisation des services de la couche peut nécessiter l'envoi d'une réponse de l'instance du destinataire à celle de l'émetteur (par exemple une quittance de réception).

Les couches communiquent entre elles au travers des points d'accès aux services (**SAP**, *Service Access Point*). C'est là qu'elles formulent leurs requêtes envers les couches inférieures. Inversement les couches inférieures peuvent spontanément annoncer des événements et passer des données à la couche supérieure (par exemple un message arrivant) : voir figure 6 en page 14.

La figure 5 en page 14 montre le détail des unités d'information qui sont passées entre les couches. Les informations sont consommées par la couche inférieure. La PDU de la couche N-1 est formée du message reçu de la couche supérieure (**SDU**, *Service Data Unit*) et des **entêtes** [15] – informations de contrôle destinées à l'instance partenaire (**PCI**, *Partner Control Information*). La même unité d'information s'appelle PDU dans la couche supérieure et SDU dans la couche inférieure.

0.2.3.4 Les types de service

Un **service** est une unité de travail qu'une couche supérieure peut demander à une couche inférieure : établir une connexion avec son instance distante, transporter des données jusqu'à cette instance distante, etc.

La réalisation d'un tel service nécessite plusieurs interactions entre la couche inférieure et la couche supérieure. Lorsque le service n'est pas confirmé, on a deux interactions : la demande d'un côté et l'indication d'exécution de l'autre. Si le service est confirmé on a alors quatre interactions – sans compter les services impliqués dans les couches inférieures pour réaliser concrètement le service.

14. cela peut signifier devoir découper le message en unités qui sont compatibles avec les couches inférieures, notamment le **MTU** de la couche 2.

15. en anglais **header** – s'ils figurent après, on peut les appeler **postètes** ou **trailer** en anglais.

L'application sur A envoie un message à l'application sur B

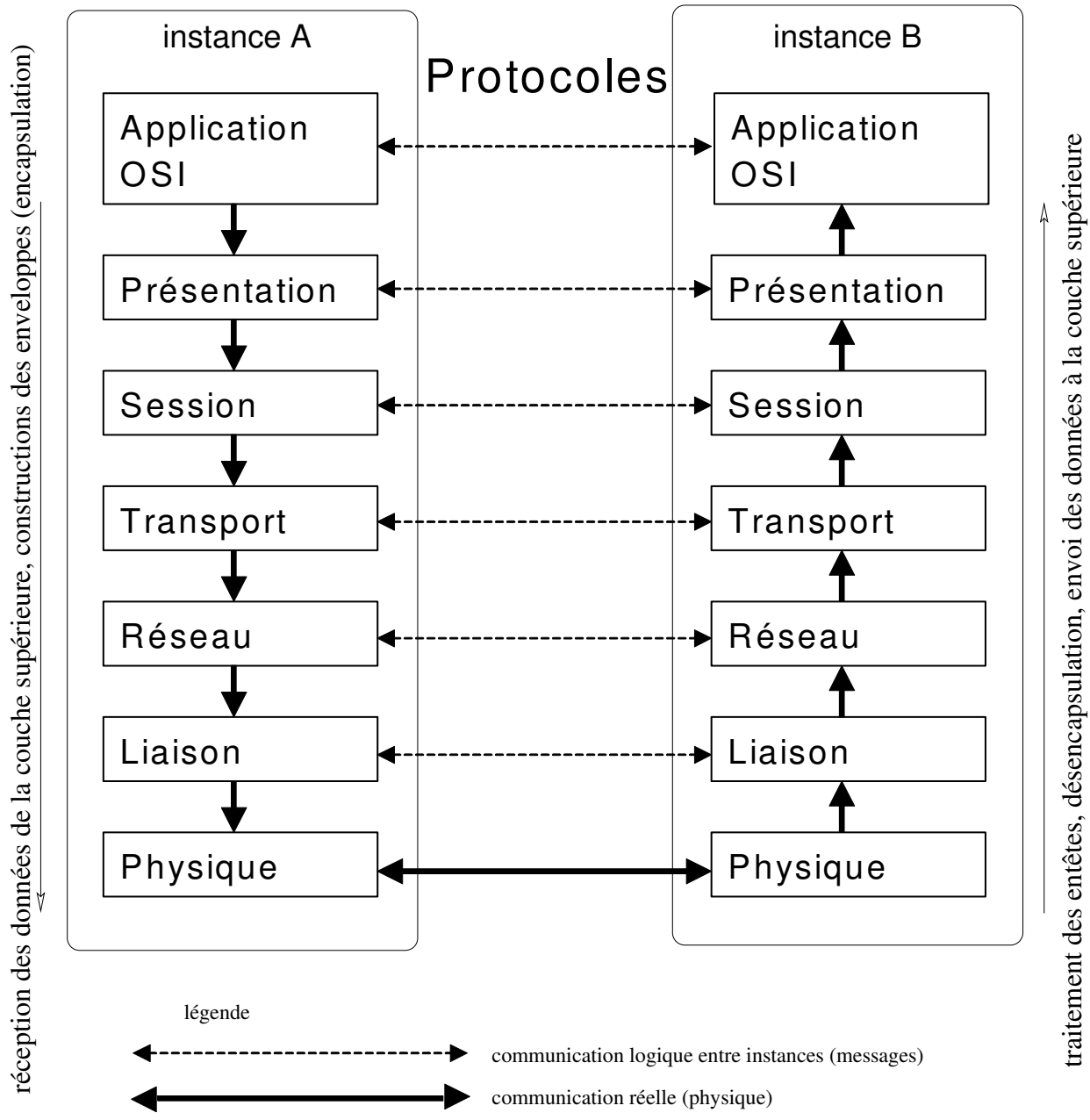


FIGURE 4 – Interaction (protocole) entre les couches

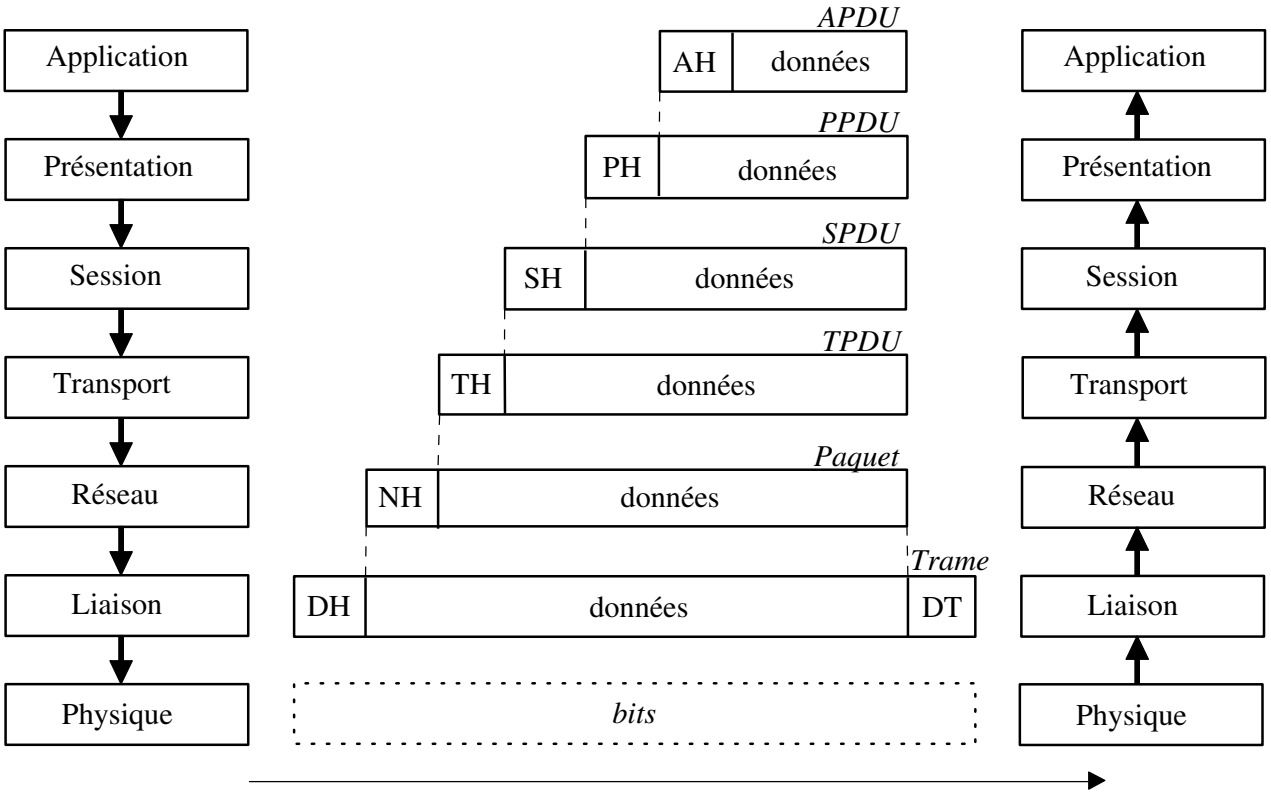


FIGURE 5 – Encapsulation – Composition et parcours effectif des unités de message (PDU)

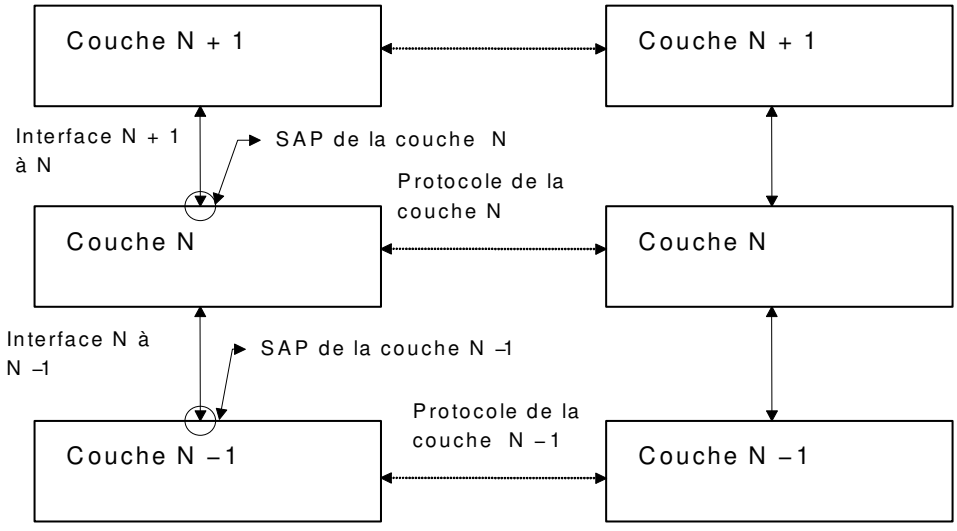


FIGURE 6 – Interactions entre les couches (services)

0.2.3.5 Les équipements

Les équipements réellement disponibles sont rarement limités à une seule couche. Cependant, pour placer les équipements réseaux dans le modèle, on considère des équipements idéaux[16] :

équipement idéal	**rôle, couche**
terminal	ordinateur, serveur, imprimante, etc : les machines qui utilisent les services du réseau : ils implémentent en général les couches 2 à 7
gateway ou **proxy**	convertisseur de protocole, serveur mandataire, relais (couche 7)
routeur	nœud qui s'occupe d'adresses et de routage (couche 3)
switch ou **commutateur**	nœud qui s'occupe d'adresses couche 2 (MAC)
répéteur	nœud qui s'occupe d'amplifier et de réémettre le signal électrique (couche 1)

FIGURE 7 – Les équipements dans un réseau

Notez qu'un **hub** ou concentrateur est un cas particulier d'un répéteur à plusieurs interfaces. De même, un **bridge** ou **pont** est un **switch** limité à deux interfaces.

En réalité, un routeur[17] cumulera souvent des fonctions de switches, voire d'autres (p.ex. **NAT**, **firewall**, interface Web de configuration...) et il existe des switches aux couches 3 ou 4, dans la mesure où les décisions de commutation sont prises en fonction d'informations figurant dans ces entêtes, p.ex. de déposer certain trafic dans un certain réseau virtuel (**VLAN**, voir section 2.4.3.2 en page 47) en fonction du numéro de port (couche 4), de l'adresse IP (couche 3) ou d'autres critères.

16. analogie : une résistance est un composant idéal : une résistance réelle a souvent aussi un effet capacitif ou inductif ; un routeur réel doit forcément être branché à une couche 2 et transmettre en couche 1, ou être configurable en couche 7.

17. attention à l'ambiguité entre le terme de passerelle couche 7 (gateway, proxy) qui est un convertisseur entre deux piles de protocoles différentes ou non directement connectées en couches inférieures, et la passerelle par défaut (parfois appelée gateway, en fait le *routeur par défaut*) que l'on configure sur un terminal, en couche 3, pour qu'il puisse sortir du sous-réseau.

0.2.3.6 Portée des couches

Comme déjà vu, les messages ne sont transmis réellement que par la couche physique. Lors de la descente dans les instances de couches en direction de la couche physique, des entêtes sont ajoutés à chaque couche (**encapsulation**). Après transmission et réception, lorsque les messages arrivent pour traitement à la couche physique, ils remontent dans les instances de couches, qui retirent – et traitent – les entêtes qui leur correspondent (voir figure 5 en page 14).

Dans certains cas, les messages ne remontent pas toutes les couches : par exemple, dans les nœuds d'un réseau (routeurs, couche 3), le message va remonter les couches jusqu'à la couche réseau et redescendre jusqu'à la couche physique. En fait, seuls les terminaux (PC, serveurs, etc) qui implémentent effectivement la couche application (couche 7) auront les messages qui traverseront toutes les couches (voir figure 8 en page 17). Concrètement, certains nœuds vont agir à la couche 3 (les routeurs), et d'autres à la couche 2 (les switches), ou d'autres uniquement à la couche 1 (répéteur).

Comme les protocoles des couches 1 et 2 *peuvent varier* d'un tronçon à un autre d'un réseau couche 3, mais que le protocole de la couche 3 sera en règle générale *uniforme* dans tout le réseau, on retrouvera des adresses à **portée** locale dans les couches 2, et des adresses à portée globale dans la couche 3.

Exceptionnellement, la couche 3 peut ne pas être identique dans tout le réseau logique : on utilisera alors de la conversion de protocole par des passerelles en couche 7 : par exemple un **proxy** pour accéder à l'Internet IPv6 depuis un réseau IPv4, ou une **gateway** pour accéder à un réseau de terrain **IoT LoRaWAN** depuis Internet.

En IPv4, les adresses utilisées en couche 3 ne sont pas toujours de portée globale : lorsque des adresses privées (voir section 3.1.3.6 en page 55) sont utilisées dans un réseau qui doit accéder Internet, on mettra en place un routeur avec fonction **NAT**/**PAT** (respectivement couche 3/couche 4) – voir section 3.2.6 en page 73.

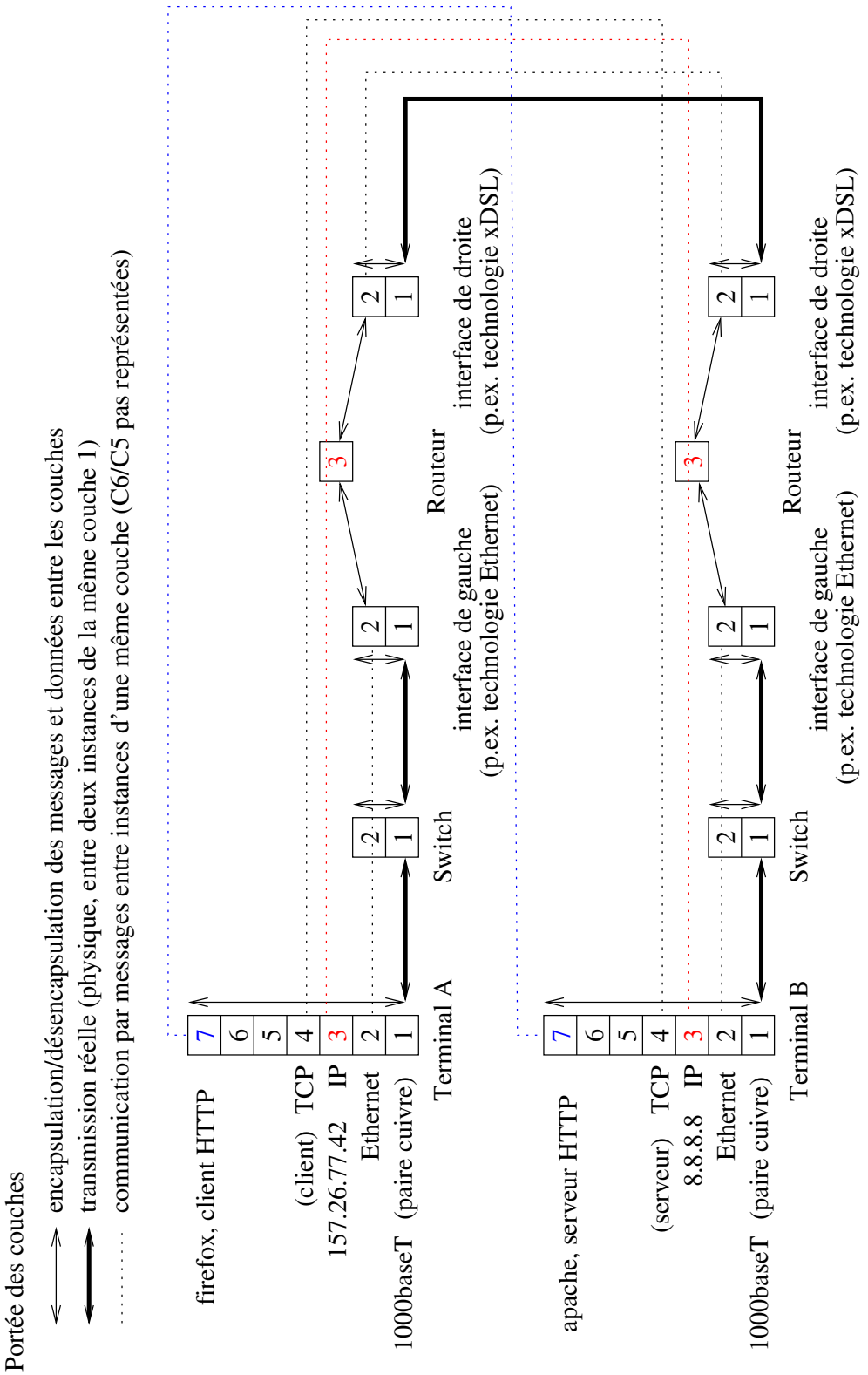


FIGURE 8 – Portée des couches

0.2.4 Le modèle Internet à 5 couches

Le modèle de référence OSI a été conçu sans tenir compte du monde Internet. Pour cette raison, on y utilise plutôt un modèle simplifié et adapté : les couches 5, 6 et 7 sont fusionnées à la couche application et les fonctions des couches **session** et **présentation** ne sont pas explicitement prévues ou normalisées.

couche	nom	description	par exemple
7	application	tout ce qui se retrouvait dans les couches 5-7 OSI	FTP, TELNET, HTTP, SMTP, DNS, NTP, SNMP, TFTP, SIP...
4	transport	transmission fiable (ou non) des données à travers un réseau	TCP, UDP
3	internet	adressage et routage dans un réseau	IP
2	network interface	assurer la transmission de données au sein d'une liaison point à point ou multipoint	Ethernet MAC, HDLC, xDSL, PPP, 802.11 (WiFi), MPLS...
1	physical	transmission physique de l'information : interface, câblage, signaux	Ethernet PHY, autres couches physiques

FIGURE 9 – Modèle IP à 5 couches

Les quatre couches inférieures ont à peu près les mêmes fonctions que leurs équivalentes du modèle OSI, à cela près que la couche 2 est plutôt vue comme l'interface à la carte réseau et devient donc une couche hybride intégrant les fonctions basses de la couche réseau classique ainsi que quelques fonctions simples de la couche liaison (typiquement et en général, uniquement de la détection d'erreur, sans retransmission).

Les couches inférieures (1-4) sont en général implémentée dans le **kernel** du système d'exploitation, contrairement à la couche 7 qui comprend l'ensemble des couches 5 à 7 du modèle OSI et est implémentée par des bibliothèques logicielles en **espace utilisateur**.

0.2.5 Schémas de réseau

0.2.5.1 Recommandations

Pour représenter des schémas de réseau, il n'est pas forcément nécessaire de suivre les conventions Cisco (voir ci-après) : on peut aussi utiliser de simples diagrammes où les équipements sont nommés (p.ex. R pour routeur, H pour hub, S pour switch), tant qu'ils sont bien documentés.

Il peut être peu clair de mélanger les couches OSI sur un même schéma en particulier pour décrire des réseaux complexes : il vaut mieux dessiner tout d'abord un schéma en couche 3 (réseau) (en explicitant les sous-réseaux, les routeurs, les adresses IP des serveurs et clients et des *lignes* abstraites), puis ensuite dessiner, si nécessaire, séparément, le schéma couche 2 (liaisons, switches, VLAN) qui peut éventuellement être complété avec des informations de la couche 1 (hubs, tableaux de brassage, technologies particulières : WiFi, PLC...) en fonction des besoins et du public concerné.

0.2.5.2 Conventions Cisco

Voici quelques exemples de représentations d'équipements selon les conventions Cisco :

routeur (couche 3)

switch (couche 2)

répéteur (couche 1)

Pour plus d'information, consulter [1], et notamment la carte de référence Cisco.

0.3 Les outils

0.3.1 Capture de trafic

Deux outils principaux seront utilisés pour la capture de trafic (couches 2 à 7) : **Wireshark** en interface graphique (GUI, plutôt pour la visualisation détaillée et la capture) et **tcpdump** (plutôt pour l'automatisation de la capture, pour une analyse rapide ou pour les systèmes embarqués) en shell. Le format de capture de ces deux outils est interopérable (basé sur **libpcap**).

0.3.2 Debugging

Les outils **ping**, **traceroute**, **mtr**, **telnet**, **netcat** (nc), **nmap**, **netstat**, **ip** (ou les anciennes commandes **arp**, **route** et **ifconfig**), **dig** et **host** seront utilisés pour le debugging réseau. L'outil **iperf** peut être utilisé pour des analyses de performance et de taux de perte.

0.3.3 Surveillance / visualisation

La visualisation de données de réseaux se base en général sur des compteurs, des captures, des journaux ou des base de données de configuration. Par exemple, le protocole **SNMP** (Simple Network Management Protocol, RFC-1157) permet de consulter voire modifier des données identifiées au sein d'un espace de nommage appelé **MIB** (*Management Information Base*). On peut donc visualiser des graphes de compteurs temps réel (p.ex. de trafic, nombre de pages imprimées sur une imprimante réseau, etc) ou traiter des alarmes. Un protocole développé par Cisco, **Netflow**, permet de mettre en place facilement des sondes sur un réseau.

Parmi les outils libres pouvant acquérir, stocker ou synthétiser ces informations, mentionnons notamment `mrtg` et **munin**, **RRDtools**, **cacti** ou encore **OpenNMS**.

0.3.4 Scannage de réseau

Pour faire la carte d'un réseau, ou pour détecter les machines qui s'y trouvent, **nmap** peut être utilisé. On peut avoir à générer du trafic dans des buts divers (test de fonctionnalité ou fiabilité, sécurité ou performance) : cet outil permet aussi de générer des trames ou paquets aux couches 2, 3 et 4. L'outil **netcat** (nc), quant à lui, peut être utilisé pour générer du trafic TCP ou UDP (couche 4).

Citons également le logiciel **OpenVAS** qui permet d'établir automatiquement des rapports de vulnérabilités de logiciels accessibles en réseau, grâce à de nombreux plugins libres ou propriétaires.

0.3.5 Génération de trafic

Pour générer ou traiter des données et entêtes par logiciel, Scapy[18] peut être utilisé, voir l'exemple ci-dessous :

Création d'un paquet IP ICMP de 192.168.0.1 en broadcast, avec deux entêtes VLAN empilés, pour tester une vulnérabilité de VLAN Hopping :

```
scapy> sendp(Ether(dst='ff:ff:ff:ff:ff:ff', src='00:01:02:03:04:05')
                    /Dot1Q(vlan=1)/Dot1Q(vlan=10)
                    /IP(dst='255.255.255.255', src='192.168.0.1')/ICMP())
```

Résultat (avec `tcpdump`) :

```
17:35:59.448896 00:01:02:03:04:05 > ff:ff:ff:ff:ff:ff,
                ethertype 802.1Q (0x8100), length 50: vlan 1, p 0,
                ethertype 802.1Q, vlan 10, p 0,
                ethertype IPv4, (tos 0x0, ttl 64, id 1, offset 0,
                   flags [none], proto ICMP (1), length 28)
```

Il ne reste plus qu'à vérifier dans quel VLAN le switch a mis la trame !

FIGURE 10 – Génération de paquets avec Scapy

0.3.6 Simulation / émulation

Comme exemple d'utilisation de la virtualisation pour émuler des réseaux complexes au sein d'une seule machine réelle, nous utiliserons le système relativement léger **Kathara**, basé conteneurs Docker.

Pour des besoins plus spécifiques, il existe également la possibilité de simuler des comportements réseaux complexes ou tester des nouveaux protocoles au sein de simulateurs 100% logiciels.

0.3.7 Virtualisation du réseau

Le réseau n'a pas échappé au *trend* de la virtualisation : les besoins des datacenters et clouds de pouvoir adresser des services sans se préoccuper de leur localisation ont mené tout d'abord à l'utilisation de **tunnel**s ou de **VPN**s, pour établir des réseaux couche 3 spécifiques, puis à une orientation service du réseau. Le protocole réseau standardisé **OpenFlow** permet de réaliser une architecture **SDN** (*Software-defined networking*)[19]. Un de ses intérêts est qu'il est composé en grande majorité de composants open source.

18. http://www.secdev.org/projects/scapy/
19. https://www.opennetworking.org/

Chapitre 1

Couche physique (1)

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison
1	Physique

Sommaire

Le but de ce chapitre est de présenter le rôle de la couche physique (en anglais *physical layer*), la couche la plus basse du modèle OSI, qui transmet effectivement l'information en la modulant de manière adéquate en fonction du support (canal de transmission), de manière fonctionnelle puis sur des exemples pratiques.

1.1 Rôle de la couche physique

La couche physique assure exclusivement le transport de flots de bits sur le support entre deux équipements communiquant par la même technologie de couche physique. Ses principales préoccupations sont liées au codage du signal et à sa lecture au bon moment (notion de synchronisation, voir section 1.2.2 en page 23), à la rapidité de modulation et à la compatibilité mécanique et électrique des interfaces.

Des systèmes de transmissions modernes et dans des conditions peu favorables peuvent toutefois ajouter de la détection d'erreur et/ou de la correction directement en couche 1, comme par exemple les codes convolutifs (treillis et turbo-codes) dans le sans-fil GSM, alors que cela est d'habitude une fonction de la couche 2 et/ou de la couche 4.

1.2 Types de transmission

1.2.1 Transmission parallèle/série

L'unité d'information traitée par un ordinateur est rarement le bit. Il s'agit plus couramment d'un ensemble de N bits, le plus souvent un octet (byte) de 8 bits ou un mot de 2, 4 ou 8 octets (word ou longword).

La **transmission parallèle** est une transmission dans laquelle les symboles binaires peuvent être émis simultanément sur plusieurs voies. Pour transmettre un octet, on émet par exemple 8 signaux sur 8 voies différentes. Ces voies physiques peuvent être indépendantes (lignes téléphoniques) ou solidaires (lignes interne d'un bus local d'ordinateur). Il est clair que le prix de revient d'une telle solution est d'autant plus élevé que l'émetteur est éloigné du récepteur. La transmission parallèle est donc uniquement adaptée aux liaisons courtes (quelques mètres), d'autant plus qu'il est difficile d'assurer, aux hautes fréquences, que l'horloge est la même pour toutes les lignes.

La **transmission série** est une transmission qui transforme les octets ou mots en une suite rythmée de bits permettant de générer un signal électrique alterné, sur une seule voie. Les états binaires 0 et 1 correspondent souvent à des tensions différentes (exemple : +V, -V). L'interprétation du signal binaire nécessite un découpage du temps en intervalles élémentaires, c'est pourquoi un signal de base de temps est en général nécessaire : on parle alors de transmission série **synchrone**.

La transmission série synchrone est celle qui, en général, est préférée aujourd'hui, car sa vitesse peut être facilement augmentée : des hybrides existent : p.ex. PCI-Express utilise une transmission série en parallèle sur un certain nombre de *lanes* (voies).

1.2.2 Transmission asynchrone/synchrone/isochrone

Lorsque la source produit des informations à des instants aléatoires, il est souvent plus simple de transmettre ces données dès leur apparition et sans tenir compte de ce qui précède, ni de ce qui va suivre. Dans cette éventualité, on a donc des trains d'impulsions ou d'informations séparées par des intervalles de temps quelconques, d'où le nom de **transmission asynchrone**. Un exemple est l'**interface série**[1] **V.24/V.28** de l'OSI (**RS-232C** de l'ANSI) : elle distingue entre terminal (**DTE** : Data Terminal Equipment) transmettant les données numériques et **modem** (**DCE** : Data Circuit-terminating Equipment) modulant les données en un signal analogique.

1. port série, port "COM"

Pour que le récepteur puisse interpréter au bon moment les signaux, la méthode habituelle consiste à compléter celles-ci avec un **délimiteur** de début et en principe un délimiteur de fin. Dans le cas d'une transmission d'octets (8 bits), on débute par un **start-bit**, permettant la synchronisation de l'horloge en phase, et on la termine par un **stop-bit** ramenant la ligne à l'état de repos, pour être prêt pour le prochain start-bit quand il arrivera.

En transmission asynchrone, il est dangereux d'augmenter la taille du paquet de données ou d'augmenter la vitesse de transmission : de légères différences de fréquences d'horloge sont inévitables : la longueur du message transmis et la vitesse doivent donc obligatoirement être limitées. En pratique, on s'arrête souvent aux 10 bits formés par 1 start-bit, 8 data-bits et 1 stop-bit, si bien que le rendement de la transmission est de 80%.

Pour éviter ces désagréments, la **transmission synchrone** a pour objet de synchroniser les partenaires, par une des quatre manières suivantes :

— par voie directe à l'aide d'une ligne réservée à l'horloge (horloge séparée) ou par une source externe de synchronisation universelle (p.ex. horloge du GPS, horloges atomiques synchronisées, etc) – coûteux, complexe
— par un codage du signal d'horloge dans le signal émis (horloge intégrée) – possible, mais coûte de la bande passante
— par un (trans)codage des données, garantissant qu'il y a suffisamment de transitions de signal – le plus usuel (par exemple **4B5B**, du brouillage, du **bit stuffing** ou l'envoi de séquences régulières, voir section 1.4.4 en page 31)

Comme la transmission est en fait continue[2], il n'y a plus besoin de délimiteurs de début ou de fin, mais le problème de la synchronisation des trames et de leurs caractères se pose quand même. Plusieurs techniques existent : une transition particulière en couche 1, une suite de bits (connus) de resynchronisation, ou d'autres techniques de couche 2.

En outre, on parle de **transmission isochrone** (iso=même) lorsque des des transmissions ont lieu à des moments réguliers (p.ex. flux audio d'une carte son USB).

Enfin, la **transmission plésiochrone** (plésio=voisin) consiste à une transmission multiplexée quasi-synchrone où l'on a prévu de la marge de fluctuation dans les phases, voire fréquences, des affluents transmis (de *plésio* signifiant voisin en grec), que l'on retrouve dans l'ancienne hiérarchie numérique plésiochrone **PDH** en téléphonie numérique classique.

1.2.3 Transmission intermittente/permanente

Les échanges de données entre ordinateurs ont des caractéristiques fondamentalement différentes de celles des conversations téléphoniques. Lorsque des personnes téléphonent, il n'est pas courant qu'elles observent de longues périodes de silence pendant leur conversation. C'est pour cela que ces types d'échanges étaient en général implémentés sous forme de **commutation de circuit**, où un circuit physique était alloué pour chaque communication au sein du réseau téléphonique analogique classique. Cela se faisait en général sous forme d'un **circuit commuté** (réseau téléphonique classique à composition).

Dans les communications entre ordinateurs, il en est tout autrement. Un ensemble de données peut avoir besoin d'être transmis très rapidement bien qu'il soit suivi d'une longue période de silence ou d'attente. Pour ce type d'échanges, la **commutation de paquets** est plus appropriée et peut éviter la complexité de l'ouverture d'une connexion. Les paquets sont de petites unités d'informations qui sont acheminées par le réseau, en général multiplexés avec d'autres usagers,

2. ou éventuellement par long train de bits synchrones, démarré de manière asynchrone et précédé et suivi de délimiteurs et d'une synchronisation (p.ex. Ethernet)

ce qui a également un avantage de **mutualisation** du coût.

Les **paquets** sont adressés à un destinataire sans s'enquérir au préalable de sa présence. La taille des paquets est fixe ou variable, mais de toute façon bornée. On utilise volontiers le terme de cellules (anglais **cells**) pour les petits paquets de taille fixe, et trames (anglais **frames**) pour les paquets de taille variable. Le chemin emprunté par le paquet pour rejoindre son destinataire peut être unique ou pas. Des commutateurs (switches) sont utilisés en couche 2 comme nœuds intermédiaires.

L'inconvénient principal du mode paquet est qu'il ne permet plus à l'usager d'être le seul utilisateur de son canal et donc que de la qualité de service (**QoS**) spécifique doit être mise en place en cas de besoin : cela peut être aussi simple qu'une priorisation de trafic sur les switches d'un LAN.

Pour d'autres formes d'échanges nécessitant un circuit permanent, on passait par un **circuit fixe**, par exemple une **ligne spécialisée** ou **ligne louée** : aujourd'hui, une application nécessitant une connexion utilisera un service de **circuit virtuel** émulé sur un réseau en mode paquet. Dans le monde Internet, le protocole de couche 4 **TCP** implémente ce **mode connecté** (de manière non permanente). Des applications complexes peuvent être supportées, y compris avec qualité de service, grâce à **MPLS** (voir section 3.2.3.4 en page 71).

1.2.4 Transmission en bande de base/à porteuse/à large bande

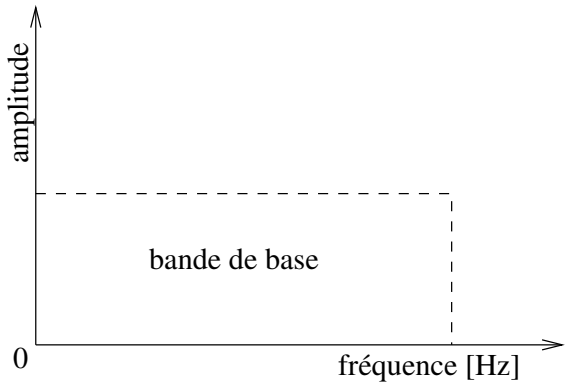


FIGURE 1.1 – Transmission en bande de base

On appelle **transmission en bande de base** (anglais **baseband**) une transmission dont le signal n'a pas subi de transposition en fréquence et peut donc utiliser toutes les fréquences du média. Ce type de transmission n'est adapté que sur des courtes distances et lorsque le support n'introduit pas d'écart de fréquence entre signaux émis et signaux reçus. Si le canal téléphonique public ne répond en général pas à cette exigence, les câbles d'installation intérieure sont tout à fait indiqués. L'avantage des systèmes en bande de base réside dans leur simplicité qui permet une mise en oeuvre économique. En effet, les dispositifs d'émission/réception ne nécessitent aucun équipement de traitement du signal avant ou après sa transmission.

Lorsque la transmission en bande de base n'est pas possible ou pas assez performante, on peut envisager une **transmission à porteuse**, ou plus précisement une modulation à porteuse, voir section 1.4.5 en page 32, en anglais **carrierband**. Cela signifie que, sur la base des données digitales, on module l'amplitude, la fréquence ou/et la phase d'un signal analogique (sinusoïdal) de référence appelé porteuse. Cette porteuse n'occupe qu'une plage de fréquence donné et peut coexister avec d'autres porteuses (en général à d'autres fréquences : **multiplexe fréquentiel**).

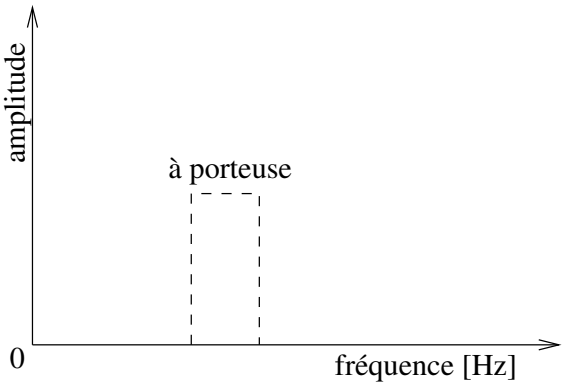


FIGURE 1.2 – Transmission à porteuse

A la différence de la transmission en bande de base, ce type de transmission permet en principe de s'adapter à la **bande passante**[3] offerte par le support, comme la bande téléphonique de 300-3400 KHz. A la réception, il s'agit évidemment de démoduler le signal analogique pour retrouver les informations sous forme binaire. Un équipement qui assure cette modulation ou démodulation est appelé **modem**.

Lorsque le support offre une très grande bande passante, on peut même penser à une **transmission à large bande** (anglais **broadband**) dont la caractéristique supplémentaire est de prévoir la subdivision de la bande passante totale en sous-canaux indépendants pour permettre la transmission simultanée de plusieurs signaux (**multiplexe fréquentiel**), ce qui a l'avantage de pouvoir résister à des perturbations ne touchant que certaines fréquences : on répartit les bits à transmettre sur des sous-canaux moins bruités (technique **OFDM** – *Orthogonal Frequency Division Multiplexing* – par exemple : câble téléréseau **CATV**, standard **DOCSYS** ; câble téléphonique en **xDSL** classique, DVB-T/C/S, DAB...).

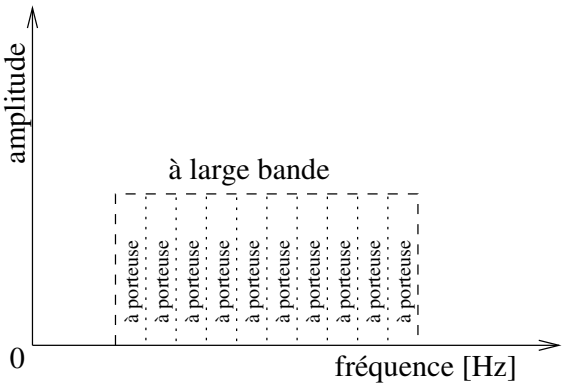


FIGURE 1.3 – Transmission à large bande (OFDM)

Une différence importante entre les systèmes de transmission à large bande et les systèmes en bande de base provient du fait que les systèmes à large bande utilisent fréquemment des amplificateurs pour régénérer les signaux analogiques : les basses fréquences ne sont pas utilisables et les amplificateurs sont unidirectionnels : pour ce dernière problème, on utilisera différentes techniques pour émettre dans les deux sens (voir section 1.2.6 en page 26).

3. plage de fréquence utilisée ou disponible en Hertz

1.2.5 Transmission analogique/numérique

Les signaux réels sont toujours, évidemment, analogiques. Toutefois, en transmission, le terme *analogique* signifie que les données originales sont sous la forme de variations continues d'une grandeur physique. Un signal analogique est donc représenté par une courbe qui évolue de manière continue au cours du temps. Les perturbations (bruit) dégraderont alors le signal transmis et l'information originale proportionnellement. Une **amplification** amplifiera également le bruit.

Aujourd'hui, on utilise plutôt de la transmission *numérique* car les données sont souvent digitales : des symboles en nombre fini (par exemple des bits valant 0 ou 1 ou des groupes de bits), obtenus par un processus de **conversion analogique-numérique**. On représentera alors les données par une transmission formée d'une suite de niveaux discrets de durée fixe. Ce signal numérique sera, comme le signal analogique précédent, perturbé par le canal. Toutefois, le problème est alors sans conséquence si le récepteur est malgré tout à même de retrouver les niveaux discrets originaux (c'est souvent le cas !) : une **regénération numérique** consistant à émettre un nouveau signal reconstruit à partir des niveaux lus approximés permettra d'annuler toutes les perturbations qui ne produisent pas de confusion [4] entre niveaux discrets. En cas de perturbation ne pouvant être supprimée par regénération numérique, la couche 4 ou 2 utilisera des codes détecteurs et une retransmission éventuelle en cas d'erreur, ou même des codes correcteurs en couche 2 ou 1, basés sur de la **redondance** structurée.

1.2.6 Transmission uni-/bidirectionnelle

Pour un système quelconque, la transmission d'information peut être à sens unique (monologue : simplex) ou bien dans les deux sens (dialogue : duplex).

Pour la transmission duplex, on fait encore la distinction entre le **full-duplex** qui caractérise une transmission simultanée dans les deux sens et le **half-duplex** (en français : duplex à l'alternat ou demi-duplex) qui caractérise une transmission alternant d'un sens à l'autre (ping-pong, question-réponse). En **liaison point à point**, la méthode la plus directe pour mettre en oeuvre le full-duplex revient à implémenter une liaison simplex pour chaque sens. Il faut donc prévoir un double câblage, comme c'est le cas pour pour l'Ethernet switché de type 100baseT (1 paire par direction).

Lorsque l'on n'a qu'une paire (2 fils), on ne peut donc en principe que faire du half-duplex. Le full-duplex peut néanmoins être réalisé grâce à diverses techniques astucieuses : la première s'inspire de la transmission à large bande (voir section 1.2.4 en page 25) en utilisant certains sous-canaux fréquentiels pour la voie descendante et d'autres pour la voie montante ; la deuxième autorise une émission simultanée sur le même support par les deux correspondants du fait qu'elle est capable de retrouver l'information utile en soustrayant à l'information reçue l'écho de l'information émise en temps réel (**suppression d'écho**). Enfin, la dernière utilise une transmission à l'alternat (half-duplex) comme le **G.Fast** DSL, sans impact significatif sur le délai, grâce à un débit élevé et un temps d'inversion très court (**multiplexe temporel**).

Une transmission duplex (ou half-duplex) peut être **symétrique** [5], ou **asymétrique** : on parle en général de transmission ou de liaison point à point asymétrique lorsque les débits montants et descendants ne sont pas les mêmes (p.ex. dans le cas de l'**ADSL**).

4. en première approche, si la différence entre deux niveaux discrets est plus grande que la demi perturbation
5. même débit dans les deux sens ; ne pas confondre avec la notion de **paire symétrique** qui signifie simplement que les deux fils ont des caractéristiques physiques et électriques très semblables : une paire symétrique est en général torsadée (toronée), de manière à offrir une immunité au bruit pour toute **transmission différentielle**.

Le problème se complique en **liaison multipoint**, y compris en sans-fil (voir section 2.3.3 en page 39).

1.2.7 Transmission simple/multiple (multiplexage)

On parle de transmission multiplexe (uni- ou bidirectionnelle) lorsqu'une voie de communication assure le transfert *simultané* de données pour plusieurs équipements **DTE**, ou simplement de plusieurs flux indépendants, en utilisant un ou plusieurs des multiplexages ci-après.

Le cas d'une transmission où les différents signaux sont modulés chacun sur une fréquence porteuse différente, porte le nom de **multiplexe fréquentiel** (**FDM** : Frequency Division Multiplexing – un exemple courant est la radio ou la télévision analogique hertzienne ou par câble téléréseau).

Un autre multiplexage est le **multiplexe temporel** (**TDM** : *Time Division Multiplexing*) qui revient à subdiviser le temps en autant d'intervalles que l'on a de communications différentes : ce dernier existe sous deux formes (la forme statique, par exemple pour certaines techniques sans-fil TDMA où chacun des canaux a son *temps de parole* propre ; et la forme complètement aléatoire, p.ex. Ethernet et ses collisions) et est clairement le type de multiplexage adapté à la transmission numérique. La capacité de transmission de la liaison conditionne le débit maximal supporté en fonction du nombre de canaux supportés. La technique de découpage fixe du temps comporte l'inconvénient majeur qu'un canal inactif bénéficie du même débit qu'un canal en pleine activité, ce qui est un gaspillage certain.

C'est la raison pour laquelle le découpage variable du temps en fonction des besoins (multiplexe temporel dynamique, statistique ou stochastique [6]) est en général préféré pour les réseaux informatiques (**mutualisation**) : il pose cependant des problèmes aux applications temps réel par son non déterminisme et à toutes les applications en raison de la non garantie d'émission.

On signalera des multiplexages faisant appels à des traitements de signaux complexes permettant l'utilisation *simultanée* des mêmes fréquences sans autre multiplexage, comme par exemple le **multiplexe par code** (**CDMA**).

Dans le domaine optique enfin, on rencontre un cas particulier de multiplexe fréquentiel : des multiplexeurs en longueur d'onde qui concentrent sur une seule fibre plusieurs flots de lumière issus d'émetteurs optiques modulant chacun un signal différent à une longueur d'onde particulière (**WDM** : Wavelength Division Multiplexing, ou multiplexe **de couleurs**). Le démultiplexage consiste alors à séparer et acheminer chaque longueur d'onde vers un détecteur optoélectronique particulier. Avec des sources lasers, capables de modulation très rapide avec une petite largeur spectrale [7], on peut ainsi espérer transmettre plusieurs dizaines de canaux allant jusqu'au Gbit/s sur une seule fibre monomodale. C'est ce qu'on appelle parfois la transmission optique à large bande.

1.3 Topologies

1.3.1 Topologies physiques

Pour connecter deux systèmes, il suffit de poser un câble de communication entre eux. On relie simplement un point à un autre, d'où le nom de **topologie point à point**. Par extension,

6. du nom des modèles statistiques nécessaires à modéliser l'**overbooking** éventuel (survente)

7. plage de fréquence en Hz, le spectre étant le diagramme en barre obtenu après passage au domaine fréquentiel par une transformée de Fourier

quand il y a plus que 2 équipements à connecter, une première possibilité consiste à relier les systèmes de proche en proche et former ainsi une topologie **chaînée**[8] de N-1 tronçons de câble interconnectant les N stations. Si la topologie en chaîne minimise la longueur totale du câblage, la rupture d'un quelconque tronçon entraîne une fragmentation en deux sous-chaînes indépendantes.

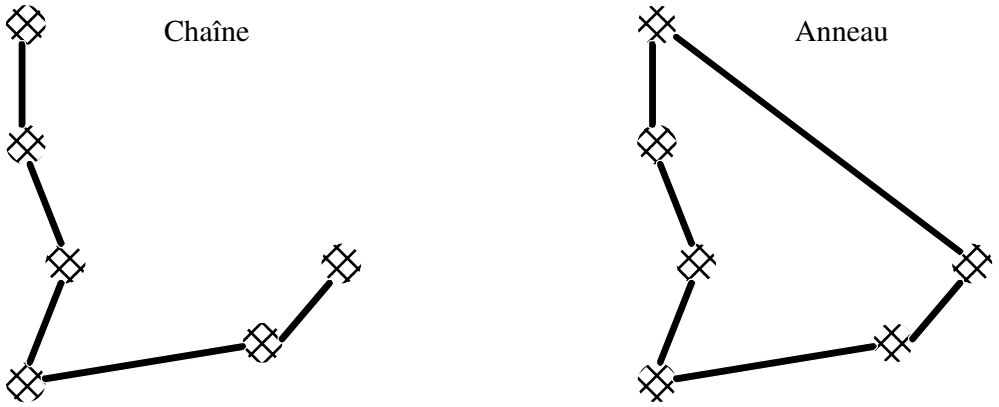


FIGURE 1.4 – Topologies chaînée et anneau

On peut aussi refermer la chaîne : dans la topologie en **anneau**, la rupture d'un câble entraîne la re-formation d'une seule chaîne. L'anneau peut donc amener de la redondance en cas de rupture, ou plus de performance en fonctionnement normal : les données pouvant être émises dans les deux directions.

Une manière très économique[9] et plus fiable que la simple chaîne pour relier des systèmes est le **bus** (appelé aussi **multipoint**) : on relie simplement les équipements à un point central, appelé **hub**, qui est un simple concentrateur-répéteur électrique (voir section 1.3.2.2 en page 30). Les différents hubs peuvent être ensuite reliés ensemble. On économise le câblage, au prix d'une moins bonne performance (multiplexage/partage du média).

Le cas le plus courant en entreprise aujourd'hui est de relier chaque station avec son propre câble à un point névralgique appelé **tableau de brassage** : on forme ainsi une topologie en **étoile** avec N câbles disposés en rayon autour d'un point central. L'avantage principal, en plus que chaque station a une liaison dédiée donc potentiellement une meilleure performance[10] réside dans le fait qu'une rupture de liaison n'affecte qu'une seule station et ne modifie en aucun cas la structure d'étoile.

En topologie **étoile**, toutefois, si le point de concentration est éloigné du centre de gravité de l'ensemble des équipements, la longueur totale du câblage s'en trouve toutefois considérablement augmentée par rapport à une topologie **bus** : un compromis est donc possible : il s'agit de créer autant d'étoiles que nécessaire et de les relier ensemble au sein de l'épine dorsale ou **backbone** de l'entreprise, avec des liaisons rapides et multiplexées : le choix des équipements reliant le backbone et au centre des étoiles est essentiel pour la performance : seul les **switch**es, des équipements de couche 2, vont pouvoir isoler le trafic des stations (voir section 2.4.1.6 en page 44) ; avec un **hub**, on reviendra aux performances d'un **bus** partagé par toutes les stations !

Dans la pratique, on rencontre souvent des topologies hybrides, par exemple bus-étoiles, bus-chaînes, étoiles-étoiles, étoiles-arbre...

8. anciennes versions d'Ethernet ; aussi SCSI : **daisy-chaining**
9. aujourd'hui, plutôt utilisé pour des applications industrielles (**bus de terrain**), par exemple pour limiter le poids du câblage : voiture, avion, etc
10. tout dépend toutefois de l'équipement central : **hub** ou **switch**, voir ci-après !

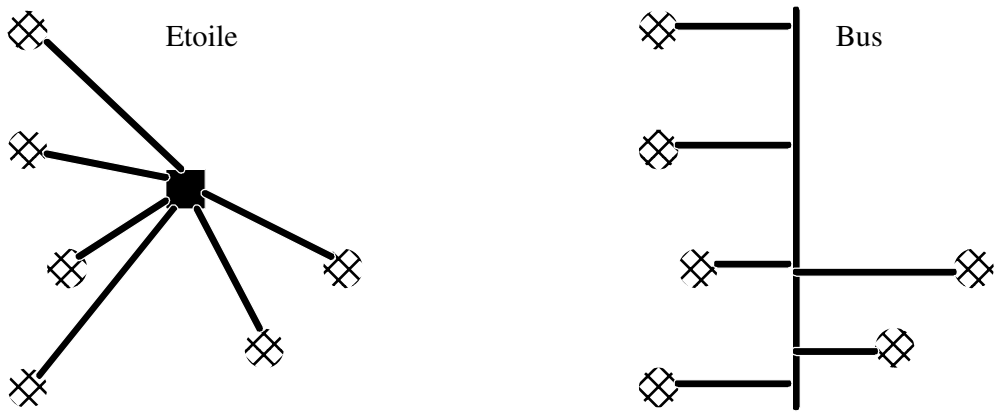


FIGURE 1.5 – Topologies étoile et bus

Lors du câblage informatique de bâtiments, c'est la topologie en étoile qui est la solution qui est aujourd'hui retenue, notamment car elle permet l'exploitation de toutes les autres topologies et de beaucoup de technologies de transmission [11]. Un ou plusieurs tableaux de brassage permettent de reconfigurer le câblage générique [12] préinstallé. Aujourd'hui, la plupart des réseaux informatiques (hors les bus industriels) sont en général disposés en topologie étoile, ou étoile-arbre, ou étoile-anneau.

Enfin, on parle de topologie en maille (**mesh**) lorsqu'il existe plusieurs chemins possibles pour relier un point à un autre. Cette redondance permet de recourir à des voies de secours en cas de défaillance sur la voie principale ou d'optimiser les échanges en utilisant simultanément plusieurs chemins pour des paquets différents. Mais, pour éviter que l'information ne tourne littéralement en rond, on doit utiliser des mécanismes sophistiqués : les développements récents permettent même la mise en place de réseaux filaires ou **WiFi** (sans-fil) auto-organisés sous forme de **mesh** (voir section 2.4.3.5 en page 48).

Dans les réseaux **MAN**, on exploite souvent une topologie en maille (avec des liaisons superflues) afin d'assurer le service même en cas de panne d'une ligne ou d'un nœud. Même les réseaux locaux d'entreprise peuvent bénéficier de configurations redondantes pour résister à des pannes (voir section 2.4.3.1 en page 46).

1.3.2 Topologies logiques

1.3.2.1 Promotion de topologies

Comme chaque câble peut contenir plusieurs conducteurs, il est possible qu'un câble de liaison assure à la fois l'aller et le retour du signal. Cette méthode permet par exemple de passer d'une topologie physique en étoile à une topologie logique en anneau (voir figure 1.6 en page 30).

Il faut donc toujours savoir de quelle topologie on parle, et à *quelle couche* : par exemple un **access point** WiFi implémente une topologie logique de type étoile (couche 2) sur la base d'une topologie bus (couche 1) **half-duplex**, car il relaie les trames entre stations et étend ainsi l'envergure du réseau.

11. s'il existe des technologies de multiplexe optique passif, elles ne sont pas utilisées dans ce contexte
12. par exemple UTP5e ou UTP6 : 4 paires cuivres

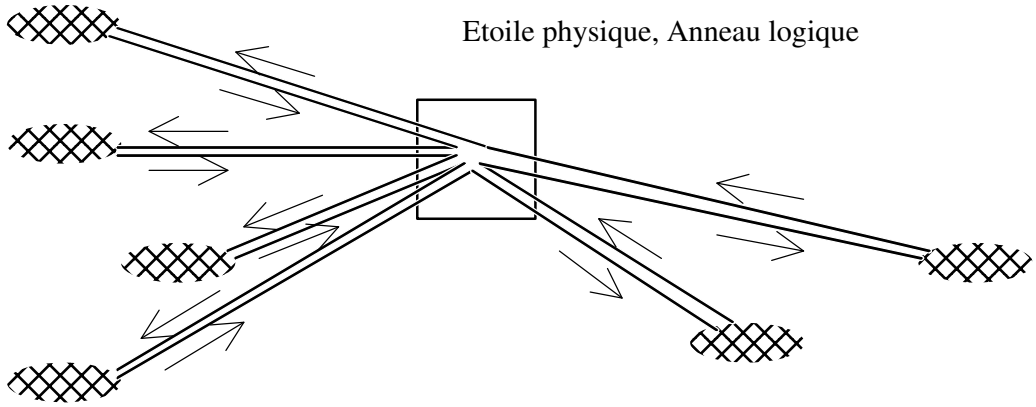


FIGURE 1.6 – Construction d'une topologie logique d'anneau sur une étoile

1.3.2.2 Exemple : LANs Ethernet

Un exemple auparavant courant dans les réseaux Ethernet était celui du **hub** Ethernet (ou concentrateur-répéteur), qui interconnectait des équipements suivant une topologie physique en étoile, mais une topologie logique de type bus (avec partage du débit entre stations et des collisions dégradant la performance, voir section 2.3.3 en page 39).

Aujourd'hui et en entreprise, l'étoile est conservée en topologie logique à l'aide du **switch** ou commutateur Ethernet, ce qui, en particulier en mode **full-duplex**, évite les collisions et garantit la performance même sur un réseau chargé, tant que le débit interne du commutateur a été bien dimensionné (voir section 2.4.1.6 en page 44). Les switches sont ensuite reliés entre eux pour former le **backbone** de l'entreprise, parfois sous forme d'un anneau performant et redondant.

1.4 Transmission numérique

1.4.1 Introduction

Dans la plupart des applications qui nous occupent, les données à transmettre sont déjà numérisées et la plupart des protocoles modernes vont les transmettre numériquement (voir section 1.2.5 en page 26). Le choix du type de transmission (voir section 1.2.4 en page 24) et de la manière dont les données sont encodées sont très importants. On différenciera ci-après les **codages de ligne** utilisés en **transmission en bande de base** des **modulations** utilisées dans les autres types de transmission – où une translation fréquentielle est nécessaire. Les choix entre codages de lignes et modulations individuels dépendront des caractéristiques du canal (quelles fréquences sont admissibles, doit-on annuler la composante continue...) et d'autres choix comme la nécessité ou non d'une horloge dans le signal, etc.

1.4.2 Bits par seconde et bauds

Lorsqu'on dénombre un certain nombre de symboles m, l'envergure de la décision binaire à prendre pour chacun de ces symboles vaut $D = \lceil \log_2 m \rceil$; ici D représente la **quantité de décision**, son unité est le **bit**. Par exemple, 8 bits encodent 256 symboles.

Le **débit binaire**, noté $\dot{D}$ (débit de décision, dérivée de D par rapport au temps) ou C (capacité

d'un canal) est le débit en bits par seconde [13] qui est disponible pour l'application – **rendement intrinsèque** (organisation des entêtes et tramage) et **rendement de protocole** (lié notamment aux délais de transmission) encore à prendre en compte.

On entend souvent l'abus de langage **bande passante** (la taille de la plage de fréquence où le signal électrique passe) alors qu'il faudrait dire **débit** (ou vitesse de transmission).

En ce qui concerne les caractéristiques physiques, comme un état particulier du canal est appelé moment, on définit notamment le **débit de moment**, noté $\dot{M}$ qui est le nombre de changements d'état du canal (p.ex. du signal électrique) par seconde, ce qui s'exprime en **Baud** ou **Bd**.

Un changement d'état du canal pouvant encoder un certain nombre de bits, la relation entre ces deux unités est exprimée par la formule :

$$\dot{D} = \log_2 m \; \dot{M}$$

avec m le nombre d'états (ou $\log_2 m$ le nombre de bits encodant les m états possibles).

Signalons que certains codages (p.ex. **Manchester**) encodent un demi-bit par moment [14], et que d'autres codages peuvent encoder plusieurs bits par moment : par exemple **PAM-4** (deux bits encodés dans un signal quaternaire – avec 4 états), ou **8B6T** (8 bits pour 6 signaux ternaires – soit à chaque moment 3^6 états possibles).

1.4.3 Critères du choix d'un code

Des critères de choix d'un code sont listés ci-après. On se rend rapidement compte qu'il faudra faire des compromis.

- pas de composante continue [15] (p.ex. pour passer les amplificateurs)
- densité spectrale faible pour des petites fréquences
- densité spectrale faible pour des grandes fréquences
- densité spectrale importante au **débit de moment** (pour récupérer l'horloge)
- peu d'états (si le **rapport signal-sur-bruit** est mauvais)

1.4.4 Transmission en bande de base : codages de ligne

Concrètement, il faut choisir des codages de ligne qui permettent la transmission en bande de base : la figure 1.7 en page 32 donne quelques exemples, avec leurs caractéristiques (retour à zéro ou non, transitions ou flancs, horloge présente ou non dans le signal, ratio $\frac{\frac{bit}{s}}{baud}$ [16]...).

Si le signal produit risque de ne pas varier et donc une perte de synchronisation pourrait en résulter, un traitement peut être appliqué aux bits pour qu'ils changent suffisamment : par exemple un vrai chiffrement, ou un brouillage (**scrambling**), qui peut se présenter sous la forme de tables de codage (**4B5B**), qui augmentent de manière fixe les signaux à transmettre. Ou du **bit stuffing** qui consiste à ajouter un signal supplémentaire de transition qui sera supprimé par le récepteur, ajoutant une incertitude sur le débit résultant. On peut aussi envoyer régulièrement des suites de bits servant uniquement à la synchronisation.

13. noté $\frac{bit}{s}$ ou encore **bps** – ne pas confondre avec des $\frac{octet}{s} = \frac{byte}{s}$, parfois noté **Bps** !
14. il y a ici deux changements d'état du canal par bit pour pouvoir récupérer l'horloge dans tous les cas
15. tension moyenne non nulle
16. représentant le nombre de bits encodant les états possibles du signal

codage	description
bipolaire	3 tensions (une positive, une négative et 0), avec retour à zéro au milieu du bit : en conséquence il y a au moins une transition par bit et donc l'horloge est présente dans le signal
Manchester	2 tensions (haute et basse), avec une transition au milieu de chaque bit : l'horloge est présente, vu que le débit de moment est forcément double du débit binaire
NRZ	2 tensions encodant directement chaque bit, pas de retour à zéro : pas d'horloge, facteur 1 : 1 ($m = 2$) entre bps et bauds, composante continue présente
NRZI	2 tensions, un changement de tension au début du bit dénote un 0, sinon 1 (pas d'horloge, composante continue) – c'est un NRZ différentiel
AMI	0 codé comme 0 Volt, 1 codé comme + ou - une tension non nulle, alternativement positive et négative (pas d'horloge, mais pas de composante continue électrique)
PAM-4	4 tensions (donc 4 états, soit 2 bits par moment), le débit binaire est double du débit de moments
MLT-3	3 tensions : on passe à la tension suivante pour un bit à 1, on reste à la même tension pour un bit à zéro : le signal varie moins que le débit binaire.
PAM-x	x tensions, codant plusieurs bits par moment, de manière différentielle pour optimiser les états représentables.

FIGURE 1.7 – Exemples de codages de ligne en transmission en bande de base

1.4.5 Modulations à porteuse

On parle de **modulation à porteuse** lorsqu'un signal est à transmettre avec une translation fréquentielle (contrairement p.ex. à une transmission en bande de base). Historiquement, il s'agissait de moduler une **porteuse** (**carrier**), par exemple un signal sinusoïdal à une fréquence fixée, en variant ses caractéristiques (amplitude, fréquence, phase). L'équipement réalisant l'opération de modulation se nomme un **modem**.

De la transmission analogique (radio AM [17] ou FM [18], télévision analogique...), nous sommes passés en bonne partie aujourd'hui à la transmission numérique (voir section 1.2.5 en page 26) : on définit un nombre d'états discrets du signal [19], et l'on change d'état à état, ce qui en anglais se nomme le *SK, Shift Keying* : voir la figure 1.8 en page 33.

Les modems utilisaient en général les modulations FSK, PSK ou APSK. Plus récemment, et cela est valable également pour les modulations les plus modernes utilisées en **xDSL**, **CATV** (DVB-C) ainsi qu'en radio DAB/DAB+ ou en télévision numérique terrestre (TNT, DVB-T) ou satellite (DVB-S), la modulation QAM est utilisée, par exemple à 64 ou 256 états. Les modulations QAM (*Quadrature Amplitude Modulation*) pouvaient être réalisées avec des signaux déphasés (en quadrature) : les versions modernes de QAM générées par DSP (*Digital Signal Processor*) peuvent être réalisées autrement, avec un gain d'efficacité.

17. modulation d'amplitude
18. modulation de fréquence
19. en général une puissance de deux

modulation analogique	modulation numérique	description
AM	**ASK**	amplitude (souvent trop sensible au bruit)
FM	**FSK**	fréquence
(PM)	(D)**PSK**	phase, éventuellement différentielle
-	**A+PSK**	phase et amplitude
-	**QAM** ou **CAP**	cas particulier de APSK, modulation dite en quadrature, pouvant être réalisée avec la combinaison de deux signaux modulés en amplitude et déphasés, représentables sur le plan de Gauss en carré (voir figure 1.9 en page 33)
-	**OOK**	tout ou rien (p.ex. lumière dans les fibres optiques)

FIGURE 1.8 – Exemples de modulations à porteuse

1.4.5.1 Phaseurs et plans de Gauss

Une représentation courante des modulations est par les **phaseurs**. On dessine sur le plan complexe (plan de GAUSS) les lieux possibles pour la modulation : l'angle par rapport à l'axe horizontal étant la phase et l'amplitude étant déterminée par la distance entre l'origine et le lieu considéré. On imagine ensuite le signal temporellement produit par la rotation du plan (phaseur).

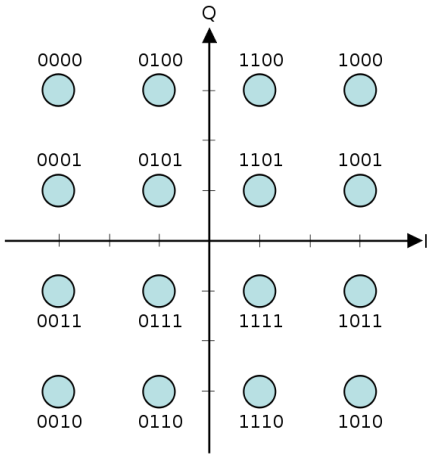


FIGURE 1.9 – Diagramme des phaseurs de la modulation QAM-16

Par exemple, une modulation QAM-16 peut être représentée comme ci-dessus (ici avec les différents états codés en **code de Gray** [20] sur le plan de Gauss [21], source Wikipedia) :

On peut se rendre compte intuitivement qu'une erreur de transmission (erreur de reconnaissance du bon lieu, ou état) se produit lorsque le **bruit** atteint la demi-distance entre deux états possibles : en conséquence, plus l'on veut encoder d'états (m augmente), plus il faut un grand rapport signal sur bruit.

20. Le code de Gray encode de manière à ce que deux états voisins ne diffèrent que par un bit, ce qui limite automatiquement les conséquences sur le nombre de bits en erreur en cas d'erreur sur un seul moment

21. Les axes Q et I correspondent respectivement aux parties imaginaires et réelles d'un nombre complexe, ou plus concrètement à un signal cosinus et un sinus modulés en quadrature et correspondant également à un déphasage de 90 degrés ou à une intégration, respectivement une dérivée.

1.5 Application : couche 1 des réseaux LAN

Les réseaux locaux d'entreprise (**RLE**, ou **LAN**) ont les caractéristiques suivantes :

- leur extension est limitée (au plus quelques kilomètres)
- un débit élevé pour des coûts bas (grâce à l'extension limitée)
- des besoins de sécurité moindre que dans les réseaux à grande distance
- un faible taux d'erreurs ($<10^{-8}$)

Il est à noter que les réseaux LAN peuvent aujourd'hui être étendus en couche 2 ou 3 par diverses techniques, comme les **VPN**s, l'utilisation de réseaux d'opérateurs comme **ATM** ou **MPLS**, ceci entre plusieurs sites sur de grandes distances (Ethernet virtuel, **Carrier Ethernet Service**), ou plus simplement entre deux bâtiments par du **WiFi** directionnel (voir section 3.2.3.4 en page 71).

1.5.1 Exemple d'Ethernet

Les principes de ce réseau couche 1 et 2 ont été développés par la société Xerox qui en construisit à la fin des années 1970 une version expérimentale à 2.94 Mbits/s fonctionnant sur un câble coaxial d'une longueur de 1000 mètres. Ce système a été appelé Ethernet en référence à l'éther, milieu à travers lequel on avait jadis pensé que se propageaient les ondes électromagnétiques [22].

Ethernet V1 vit donc le jour en 1980 avec un débit nominal de 10 Mbits/s, suivi en 1985 par la version Ethernet V2 (Ethernet II, aussi appelée DIX, du nom des sociétés DEC, Intel et Xerox qui l'ont conçue), avant sa normalisation par l'IEEE dans sa recommandation 802.3. D'une manière générale, la norme prévoyait initialement une transmission bidirectionnelle à l'alternat (half-duplex) de trames asynchrones en transmission synchrone, basée sur un codage numérique de type **Manchester** (débit binaire de 10 Mbits/s => modulation à 20 Mbauds => horloge de 10 MHz).

1.5.1.1 IEEE 802.3 PHY (couche physique)

1.5.1.1.1 Anciennes normes Ethernet Ces normes sont historiques et reposent sur un réseau Ethernet classique (bus avec collisions et **CSMA/CD**), avec des résistances de terminaison de 50 Ohms :

norme	description
10base5	débit à 10 MBit/s, topologie bus (quasi chaîné), câble coaxial épais jaune (**thickwire**), 500 m de longueur maximum, la connexion d'un nouveau tronçon ou d'un terminal peut se faire sans débrancher (**transceiver vampire**).
10base2	câble coaxial fin noir (**thinwire**), 185 m maximum, dérivation par prise en T (BNC), limites au nombre de stations possibles.
10baseT	**transmission différentielle** sur paire torsadée (**paire symétrique**) non blindée (UTP : Unshielded Twisted Pair) à 4 fils (connecteur **RJ45 8P8C**, sens émissions et réception séparés chacun sur une paire) : topologie physique en étoile, chaque station étant reliée à un concentrateur-répéteur (**hub**) pour former une topologie logique de type bus.

22. en l'espace libre : Albert EINSTEIN a montré au début du 20^{e} siècle que l'éther n'existe pas, ce qui a eu de nombreuses conséquences : la vitesse de la lumière c est une limite et les mouvements relatifs provoquent des variations infimes du temps, dont il faut tenir compte par exemple dans le **GPS** ; dans les matériaux, la lumière a une vitesse de propagation inférieure, par exemple dans les fibres optiques usuelles $\frac{2}{3}c$ environ

1.5.1.1.2 Fast Ethernet Il s'agit de trois normes (IEEE 802.3u) qui ont fait évoluer le concept originel d'Ethernet vers un débit nominal décuplé de 100 Mbps. Conformément à la tendance actuelle du câblage structuré, seules la paire torsadée et la fibre optique ont été retenues.

La norme **100BaseTX** est prévue comme une extension directe de 10BaseT et prévoit ainsi d'utiliser deux paires d'un câble UTP-5 avec une connectique toujours de type RJ-45 sans blindage. Les interfaces performantes sont capables de commuter automatiquement entre 10 et 100 Mbps grâce à la technique **autosense** (voir section 1.5.1.1.6 en page 36). La transmission peut avoir lieu en mode full-duplex, une direction par paire. Le codage de ligne est **MLT-3** précédé d'une table de codage **4B5B** permettant de garantir des transitions.

La norme **100BaseFX** préconise l'utilisation d'une paire de fibres optiques multimodales exploitée en première fenêtre (850 nm) sur une distance maximale de 200 mètres. Grâce au mode full-duplex qui abroge la limite du **round-trip time** (temps aller-retour), cette distance peut être portée à plus de 1000 mètres.

La plupart des équipements supportent aujourd'hui au moins 100BaseTX, en full-duplex, et la topologie mise en place est en étoile physique et logique (réseau Ethernet switché, dont les switches principaux sont interconnectés sur un **backbone** en général plus rapide). En conséquence, il n'y a plus de collisions : le protocole **CSMA/CD** (voir section 2.3.3 en page 39) n'est plus nécessaire puisqu'une station peut émettre et recevoir simultanément. S'il y a surcharge, c'est aux switches de gérer le contrôle de flux (voir section 2.4.1.6 en page 44).

1.5.1.1.3 Gigabit Ethernet Cette norme à 1000 Mbps (IEEE 802.3z en fibre, IEEE 802.3ab en cuivre) est devenu la norme de facto de l'Ethernet actuel, que l'on retrouve désormais partout sous sa variante cuivre (alors que le **backbone** passe en 10 GBit/s fibre voire cuivre, voir prochain paragraphe). Les variantes suivantes sont définies :

1000BaseLX laser grandes ondes (1300nm) sur fibre optique multimode et monmode, jusqu'à 3km

1000BaseSX laser ondes courtes (850nm) sur fibre optique multimode, jusqu'à 500m

1000BaseCX câble en paires torsadées blindées 150 Ohm pour connexions entre serveurs, jusqu'à 25m (aussi utilisé par Infiniband)

1000BaseT câble en paires torsadées (4 paires, soit 8 fils) non blindées de catégorie 5 au moins, jusqu'à 100m – c'est le standard le plus courant

1000BaseTX câble à 2 paires torsadées UTP6

La partie physique (connecteur, fibre) des variantes sur fibre optique s'appuient sur le standard **Fibre Channel** ANSI X3T11, tandis que la partie MAC est inspirée de la spécification IEEE 802.3 (voir section 2.4.1 en page 39).

Le **1000BaseT** (8 fils UTP5/5e, 4 paires), **full-duplex**, le plus courant en cuivre, a amené quelques simplifications : p.ex. un seul niveau de répéteur (jamais utilisé), mais aussi des améliorations : groupement de trames, autocroisement des câbles[23]. La transmission utilise un brouillage[24] puis transmission en parallèle sur les 4 paires en **PAM-5**. Ici, les 4 paires encodent $5^4 = 625$ états à moment du signal, alors que seuls 8 bits utiles sont nécessaires pour les données, correspondant à 256 états. Des états supplémentaires, 256 permettent d'encoder

23. les 4 paires étant utilisées à la fois en émission et en réception, grâce à de la suppression d'écho performante
24. chiffrement avec une séquence pseudo-aléatoire connue – seules les versions fibre, ou cuivre sur 25 mètres 1000BaseCX utilisent des tables de codage **8B10B**

de la correction d'erreur (**FEC**, *Forward Error Correction*) par un code convolutif et le reste sert au contrôle [25].

1.5.1.1.4 10-Gbit Ethernet Le 10-GBit Ethernet est souvent utilisé pour le **backbone**. Il existe en variante fibre (longues distances p.ex. 20 km, utilisée déjà même entre deux bâtiments voisins ou en cas de problèmes d'immunité au bruit), ou en cuivre : les variantes cuivres diffèrent par la longueur possible (15m de câbles spéciaux pour les premières normes comme **10GBase-CX4** ; de 55 mètres à 100 mètres suivant le type de câblage respectivement catégorie 6 ou 6a/7 pour la norme **10GBaseT**, qui utilise les connecteurs classiques RJ-45 8 fils, s'affranchissant des transceivers SFP+, QSFP/QSFP+).

1.5.1.1.5 Full-duplex Ethernet Comme déjà mentionné, dès l'instant où chaque station est desservie par son propre segment (topologie étoile vers un switch), avec une paire pour la transmission, et une paire pour l'émission la problématique des collisions (**CSMA/CD**) devient en quelque sorte caduque. Par exemple, sur un 100baseT à 2 paires (4 fils), une paire sert à l'émission, et une paire à la réception.

1.5.1.1.6 Auto-négociation Ethernet (autosense) La norme IEEE 802.3u prévoit la possibilité de négocier les paramètres de la liaison (débit, **full-duplex**/**half-duplex**, **contrôle de flux**, ou autocroisement [26]), ceci dès 10baseT (optionnellement), et obligatoirement dès Gigabit Ethernet (1000baseT, cuivre).

Cette auto-négociation permet à deux périphériques interconnectés de définir les paramètres de la liaison en choisissant les meilleures paramètres compatibles entre les équipements.

Elle repose sur une modification des impulsions utilisées par les terminaux en 10baseT pour vérifier la présence du lien (**heartbeat**).

Il est toujours possible de forcer les équipements dans un mode ou l'autre, mais cela peut avoir pour conséquence des incohérences de modes et une performance dégradée.

25. similairement aux codes non affectés aux données de 4B5B/8B10B non utilisés ici, voir https://www.iol.unh.edu/sites/default/files/knowledgebase/ge/pcs.pdf p.53, p.71

26. la connexion directe de deux ordinateurs en 100baseT nécessite un câble dont les paires d'émission et de réception sont croisées, sans autonégociation MDI-X

Chapitre 2

Couche liaison (2)

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	**Liaison**
1	Physique

Sommaire

Le but de ce chapitre est de présenter le rôle de la couche liaison (en anglais *link layer*), qui assure certaines fonctions de délimitage de trames, d'intégrité, de contrôle d'accès au média et, si nécessaire de retransmission, de manière fonctionnelle puis sur des exemples pratiques.

2.1 Rôle de la couche liaison

C'est la couche de liaison de données qui détermine la façon dont les bits venant de la couche physique sont regroupés en trames et se charge de traiter les erreurs de transmission. Elle définit aussi la méthode d'accès au support, c'est-à-dire qu'elle orchestre la communication pour l'ensemble des participants (en particulier sur une **liaison multipoint**) et régularise le volume des données échangées entre entités source et destination. Finalement, elle offre une interface de service clairement définie à la couche réseau. Cette couche liaison est traditionnellement découpée dans les réseaux locaux en deux demi-couches distinctes :

la demi-couche **LLC** (*Logical Link Control*) qui s'occupe notamment du traitement en cas d'erreurs – n'est *pas* présente dans les piles de protocoles IP où son rôle est plutôt joué par la couche transport (4)
la demi-couche **MAC** (*Media Access Control*) qui traite des problèmes spécifiques de tramage, de l'adressage des stations, de la détection d'erreurs (sans retransmission dans les piles IP) et du contrôle d'accès au support

2.2 Tramage

Du fait que les trains de bits émis peuvent être reçus par la couche physique avec des erreurs affectant les valeurs binaires ou même le nombre de bits, le découpage en trames des flots de bits est plus difficile qu'il n'y paraît à première vue. La première possibilité qui consiste à insérer des temps morts pour délimiter les trames n'est guère réalisable puisque les réseaux ne garantissent en principe pas les délais – sans parler du gaspillage introduit. On utilise donc couramment plusieurs autres méthodes orientées sur les caractères ou sur les bits :

— faire précéder chaque trame d'un champ contenant le nombre total de caractères : cette méthode utilise un champ additionnel dans l'entête de la trame pour indiquer le nombre d'octets qu'elle contient. Un problème crucial se pose lorsque ce champ est corrompu au cours de la transmission. Même si le récepteur peut s'en rendre compte (grâce à une somme de contrôle : **checksum**, **CRC**...), il ne peut pas savoir où commence réellement la trame suivante : une perte totale de synchronisation peut en résulter.

— délimiter le début et la fin de chaque trame avec des symboles de contrôle (sous forme d'un **fanion** de bits spécifiques), ce qui résout le problème de la synchronisation après une erreur de transmission ; reste celui de la transparence des données : ces symboles spéciaux ne peuvent figurer dans les données : une solution est l'**échappement** par un symbole d'échappement qui doit lui-même être doublé [1] lors qu'il est utilisé dans les données, une autre est d'assurer que les symboles ne puissent jamais figurer dans les données (par exemple avec Ethernet, des codes interdits d'une table de codage **4B5B**, en les agençant de manière à ne pas perdre la synchronisation de couche 1 ; ou alors par l'ajout si nécessaire d'un bit dit de transparence à l'endroit propice (**bit stuffing**), qui est automatiquement retiré par le récepteur qui peut alors reconnaître sans équivoque les fanions de délimitation de trames – **HDLC** insère par exemple un `0` de transparence après cinq `1` consécutifs pour préserver le fanion `01111110`).

— transgresser ou violer le codage utilisé dans la couche physique : cette dernière méthode de délimitation est employée lorsque le codage sur le support physique contient des redondances. Le code **Manchester** représente par exemple le bit `0` par une impulsion positive suivie d'une impulsion négative (front descendant) et vice versa pour le bit `1`. Les combinaisons positive-positive et négative-négative ne sont pas prévues par le codage. Certains protocoles (RNIS, 802.5) utilisent une séquence invalide telle que deux impulsions positives suivies de deux impulsions négatives pour délimiter leurs trames.

Pour être complet, on peut ajouter que certains protocoles s'inspirent de méthodes combinées : par exemple 802.3 définit un fanion de bits comme préambule et comme postambule et un champ de longueur pour permettre d'anticiper la fin d'une trame ; au contraire, Ethernet classique, qui exploite ce champ comme un type, dépend du fanion postambule pour répérer la fin de trame : ce fanion est formé de deux codes de 4 bits interdits, mais dont l'emploi n'est pas risqué, de la table de codage **4B5B**, et la longueur est aussi spécifiée en couche 3 IP.

1. **HDLC** asynchrone utilise les codes ASCII DLE STX et DLE ETX comme délimiteurs, et DLE DLE dans les données lorsque le code ASCII DLE doit être transmis

2.3 Accès multiple au canal

2.3.1 Partage du canal

Lorsque le canal est partagé[2] entre plusieurs équipements (topologie multipoint ou bus), ou que les deux sens de transmission ne disposent pas de leur canal propre (half-duplex), le choix de la méthode d'accès au canal est essentielle.

2.3.2 Mode déterministe

Une possibilité est d'assurer un partage *déterministe* – prévu à l'avance de manière statique – du canal : que cela soit par **scrutation** (anglais : **polling**, voir aussi la section 0.2.2.2 en page 7) des différents équipements esclaves par un équipement maître, ce qui est souvent utilisé sur les réseaux de terrain / bus industriels, ou par une répartition temporelle fixe le moment récurrent où chaque équipement peut parler (**multiplexe temporel statique**, **TDM** statique).

Ce mode a bien sûr l'avantage d'assurer une interrogation régulière, connue par avance, des équipements, avec un grand problème en cas de panne du maître[3] et en général un risque de gaspillage du temps et donc du débit total résultant lorsque les stations n'ont rien à communiquer.

2.3.3 Mode aléatoire ou à collisions

L'autre est de définir une méthode d'accès à collisions, permettant aux stations d'émettre à peu près quand elles le *nécessitent*, tout en assurant un accès au canal le plus fiable possible. On regroupe ces méthodes sous l'acronyme **CSMA** (*Carrier Sense Multiple Access*, accès multiple à détection de porteuse ou de signal).

Ces méthodes diffèrent dans leurs caractéristiques[4] :

Voir la section 2.4.1.2 en page 40 pour plus de détails dans le cas concret d'Ethernet.

2.4 Exemple d'Ethernet

2.4.1 IEEE 802.3 MAC

2.4.1.1 Introduction

La norme IEEE 802.3, adoptée par l'ISO sous la référence IS 8802.3, appartient à la famille des réseaux locaux de type CSMA/CD 1-persistant. L'abréviation anglaise **CSMA/CD** (*Carrier Sense Multiple Access/Collision Detection*) signifie en français *Accès multiple avec écoute de la porteuse et détection de collision* (voir section 2.4.1.2 en page 40). Le fonctionnement de cette méthode d'accès repose sur la connaissance a priori d'un délai maximal de propagation entre les deux stations les plus éloignées du réseau. Dans le cas qui nous occupe, ce délai maximal est fixé à 25.6 us, ce qui signifie que le signal doit toujours pouvoir effectuer un aller-retour entre les deux stations les plus éloignées en moins de 45 us (round-trip delay) pour avoir une marge par rapport à la limite théorique de 51.2 us.

2. sans multiplexage séparant les canaux, comme p.ex. du multiplexage fréquentiel.
3. il existe des variantes sans maître centralisé, comme les réseaux à jetons
4. CSMA/CA devient CSMA/CD en cas de collision avérée

méthode	nom	caractéristiques	exemple
CA	collision avoidance	éviter les collisions par réservation préalable pour les longues trames	WiFi 802.11
CD	collision detection	en cas de collision, réémission après un délai aléatoire croissant : peut mener à une saturation rapide du canal	Ethernet classique
CR	collision resolution	résoudre les collisions par exemple par une priorité fixée des stations (SCSI interface parallèle – obsolète) ou des caractéristiques de la trame envoyée (*bitwise arbitration*, CSMA/BA), grâce à un écho des bits reçus (ISDN) ou à un câblage particulier faisant que les 0 écrasent les 1 (CAN) ; attention : sans précautions particulières, peut mener à une **famine**	bus CAN, I2C (arbitration), SCSI, ISDN

FIGURE 2.1 – Les différents modes CSMA

2.4.1.2 CSMA/CD Ethernet et gestion des collisions

A l'origine, les réseaux de type Ethernet sont basés sur une **topologie bus**. A un canal de communication commun sont attachées plusieurs stations. Chaque station possède une adresse unique (l'**adresse MAC**) et peut ainsi reconnaître les trames qui lui sont destinées et ignorer les autres[5]. Il est clair que l'accès au bus doit être réglé (voir section 2.3.3 en page 39). Si deux stations ou plus émettent en même temps, il y aura **collision** et les trames seront incompréhensibles.

On utilise le procédé suivant :

1. **carrier sense** (CS) : une station qui veut émettre *écoute* le canal afin de savoir s'il est momentanément libre. Si ce n'est pas le cas elle reste à l'écoute jusqu'à que la voie se libère durant au moins 9.6 us (caractère *persistant*).
2. si le canal est libre elle **émet** sa trame (pour une durée minimale de 51.2 us – ce qui correspond aux 64 octets de la taille minimale classique de trame Ethernet)
3. **multiple access** (MA) : malgré l'écoute du canal, il se peut que deux stations se mettent à émettre simultanément car la propagation du signal a une vitesse limitée
4. **collision detection** (CD) : chaque station qui émet contrôle parallèlement à l'émission si d'autres stations émettent, ce qui modifierait considérablement les caractéristiques de son signal. Si plusieurs stations émettent, on dit qu'une collision a lieu, et toutes les stations cessent d'émettre[6]
5. **exponential backoff** : chaque station impliquée dans la collision attend une durée aléatoire (multiple entier de 51.2 us[7], pris dans l'intervalle [0 .. 2^n - 1], avec n le nombre de collisions consécutives[8]) et réessaye d'émettre (retour au point 1)

5. les trames **broadcast** et **multicast** doivent être traitées également – voir section 3.1.4 en page 56
6. après avoir accentué la collision par une émission spéciale : une séquence de brouillage de 3.2 us pour avertir les stations qui n'auraient pas réagi
7. soit deux fois le délai maximum de propagation entre deux stations
8. sans dépasser 1023 fois 51.2 us pour le délai, en abandonnant la transmission après 16 collisions

Comme le réseau a une extension physique maximale définie par les normes, les collisions ne peuvent avoir lieu qu'au début de l'émission d'une trame. Il est important que les collisions soient toujours détectées par toutes les stations avant la fin de la réception de la trame la plus courte.

Le temps aléatoire d'attente après une collision est généré de façon à ce qu'en moyenne, il augmente à chaque nouvel essai (**exponential backoff**). A chaque collision une partie de la capacité de la ligne est perdue. Plus le trafic est grand plus le nombre de collision augmente. De ce fait on ne peut utiliser plus de 60% à 70% de la capacité.

Toutefois, aujourd'hui, Ethernet (10/100/1000baseT+), est presque toujours exploité en topologie étoile en couche 1 et couche 2 (switches) et en **full-duplex** : en conséquence, *les problèmes de collisions ne se posent plus*. Sauf dans des cas particuliers[9] on peut donc considérer aujourd'hui que le problème principal est le débit interne total de chaque switch ainsi que la performance globale du **backbone** interconnectant les switches de l'entreprise, ainsi que le le délai ou la variance de délai (**jitter**) sur les liaisons chargées ; pour ce dernier problème, on peut, sur les switches, limiter le trafic avec le **flow control** et prioriser certains trafics p.ex. avec **802.1p** sur le backbone (voir section 2.4.3.2 en page 48).

2.4.1.3 Format des trames

2.4.1.3.1 Trames 802.3 Le détail du format des trames 802.3 est le suivant (taille exprimée en bits, champs IFG et PA+SFD ne faisant pas partie de la couche 2) :

IFG	PA + SFD	DA	SA	LEN	INFO (+PAD)	CRC
96	62 + 2	48	48	16	368-12000	32

2.4.1.3.2 Trames Ethernet Les trames Ethernet classiques[10] possèdent un champ **TYPE** à la place de **LEN** :

IFG	PA + SFD	DA	SA	Type	INFO (+PAD)	CRC
96	62 + 2	48	48	16	368-12000	32

Le champ virtuel IFG (*InterFrame Gap*) correspond au temps de calme minimal de 9.6 us après lequel l'émission peut véritablement commencer. Le préambule PA est plutôt du ressort de la couche physique puisqu'il contient une alternance d'au moins 62 bits `1010..` pour permettre la synchronisation bit (reconstitution du rythme d'horloge). Le délimiteur de début SFD (*Start of Frame Delimiter*) est formé par la séquence de bits 11 qui permet la synchronisation (l'alignement) au niveau octet. On peut aussi voir cela comme 7 octets PA 0x55 (56 bits) suivis d'un octet SFD 0xD5 (8 bits), soit 8 octets ou 64 bits au total.

L'entête véritable peut donc commencer : au tout début (optimisation, voir section 2.4.1.6 en page 44), on trouve l'adresse MAC de destination DA suivie par l'adresse MAC de la source SA. On a pour habitude de représenter ces adresses de 48 bits par 6 paires de chiffres hexadécimaux séparées par des tirets ou des deux-points (exemple : `08-00-2B-43-A9-0B` ou `08:00:2b:43:a9:0b`). La règle veut que le bit le moins significatif (**LSB**, *Least Significant bit*)

9. par exemple **bus de terrain**, pour des applications industrielles par exemple
10. Ethernet II (ou DIX, voir section 1.5.1) – toujours utilisées par IP !

du premier octet de DA envoyé permette au récepteur de faire la distinction immédiate (premier bit reçu !) entre une adresse individuelle (**unicast** : LSB=0) et une adresse de diffusion [11] globale (**broadcast**) ou restreinte/de groupe (**multicast**) : LSB=1).

broadcast/multicast	**type (hexadécimal)**	**description**
FF-FF-FF-FF-FF-FF	0800	IP Broadcast (RFC-919) – voir section 3.1.4.2 en page 57
	0806	ARP Broadcast
	8035	RARP Broadcast
	809B	EtherTALK Broadcast
01-00-5E-xx-xx-xx	0800	IP Multicast (RFC 1112, section 6.4) – voir section 3.1.4.3 en page 57
01-80-C2-00-00-0x	(length)	Spanning Tree Multicast (voir section 2.4.3.1 en page 46)
01-80-C2-00-00-14	(length)	OSI Route Level 1 (within area)
09-00-2B-00-00-0x	xxxx	DEC LAT/LTM/NetBIOS...
09-00-2B-01-00-0x	8038	DEC LanBridge (Hello packets)
09-00-2B-02-01-0x	803x	DEC DNA
80-01-43-00-80-0x ?	(length)	FDDI (MSB first !)
C0-00-00-00-00-0x ?	(length)	Token Ring (MSB first !)
FF-FF-00-x0-00-0x	81D6	LANtastic

FIGURE 2.2 – Adresses et types Ethernet pour broadcast ou multicast

Pour que chaque adresse SA soit universellement unique, l'IEEE se charge d'octroyer une ou plusieurs combinaison(s) différente(s) des 3 premiers octets (**OUI**, *Organization Unique Identifier*) à chaque fabricant, qui dispose alors librement des 3 derniers octets (2^{24} = 16'777'216 adresses). Le bit juste avant le LSB du premier octet indique en outre le caractère global ou local (0/1) de l'adresse.

Puis, on trouve un champ de longueur LEN(gth) codant sur 16 bits le nombre d'octets du champ d'information. Ce nombre est par définition inférieur ou égal à 1500 [12] (0x05DC) – cela permet de différencier les trames au format Ethernet classique [13], où ce champ de 16 bits est exploité pour le codage du type de protocole (codé par un entier supérieur à 1500, voir figure 2.3 en page 43) utilisé par la couche supérieure. Les deux variantes de protocole Ethernet et 802.3 peuvent donc cohabiter *simultanément* sur un réseau : les récepteurs d'une trame peuvent *déterminer* s'il s'agit de 802.3 ou d'Ethernet grâce à la séparation entre domaines de valeur du champ type/longueur.

Si tout à coup la longueur d'une trame est inférieure à 46 (0x2E), la norme prévoit de **bourrer** (*padding*) le champ d'information jusqu'à atteindre 46 octets, de façon à garantir que la trame complète contienne au moins 64 octets (512 bits) pour garantir la détection correcte des collisions avec la limite usuelle de l'étendue maximale du câblage jusqu'au prochain switch (100m).

Avec des trames au format 802.3, on est droit de se demander où l'information indispensable du type de protocole de niveau supérieur est codée : un entête spécifique est placé dans les

11. le champ Type est de plus utilisé (voir figure 2.2 en page 42)
12. les tags de VLAN (voir section 2.4.3.2 en page 47) augmentent cette limite de 4 octets et les **jumbo frames** peuvent aller jusqu'à 9000 octets, mais ces deux exceptions sont des trames Ethernet et pas 802.3
13. toujours utilisées dans le monde IP !

type (hexadécimal)	protocole
0600	Xerox XNS
0800	Internet Protocol version 4 (IPv4)
0805	X.25 Level 3
0806	Address Resolution Protocol (ARP)
3C0x	3COM NBP (netBios Protocol)
8035	Reverse ARP (RARP)
8037	Novell IPX
8100	Trame tagguée 802.1q (voir section 2.4.3.2 en page 47) et norme IEEE 802.1aq
814C	SNMP over Ethernet (RFC-1089)
86DD	Internet Protocol version 6 (IPv6)
880B, 8863, 8864	Point-to-Point Protocol (PPP) et PPP over Ethernet (PPPoE)
8847 8848	MPLS (voir section 3.2.3.4 en page 71) MPLS multicast
88A2	ATA over Ethernet
8906	Fibre Channel over Ethernet
9000	Loopback (Configuration Test Protocol)

FIGURE 2.3 – Quelques types de protocoles transportés par Ethernet

données utiles (juste au début du champ d'information présent dans ces différentes trames – voir la section 2.4.2 en page 45 sur 802.2 LLC/SNAP).

2.4.1.4 Fiabilité / gestion des erreurs

Tel que décrit jusqu'à présent, le protocole **CSMA/CD** (voir section 2.4.1.2 en page 40) ne fournit aucune information sur le sort subi par une trame transmise sans incident apparent. L'absence de collision ne garantit pas que la trame est transmise au destinataire sans aucune altération. Pour que la transmission puisse être considérée comme étant réussie, il faut que le destinataire vérifie que tout s'est bien passé en contrôlant [14] le CRC situé en fin de trame. Si le reste de la division est nul, le récepteur conclut à la bienfacture (vraisemblable) de la trame reçue, sinon il ignore la trame en erreur.

De toute manière, certaines trames peuvent carrément être perdues par des nœuds comme les ponts ou les commutateurs (switches) : le cas de la perte est géré, si désiré et en fonction des besoins de l'application, dans les couches supérieures (p.ex. couche 4 TCP dans le cas usuel où Ethernet est utilisé, ou dans LLC types 2 et 3 si 802.3 est utilisé).

2.4.1.5 Réseaux couche 2 étendus (ponts)

2.4.1.5.1 Rôle principal d'un pont Pour étendre le réseau du point de vue de la distance (et/ou du nombre de stations), il faut recourir à des ponts (*bridge*, niveau 2 OSI) pour interconnecter deux réseaux physiques différents.

14. il s'agit ici d'effectuer la division modulo 2 de la trame réceptionnée par le polynôme générateur de la redondance cyclique, l'AUTODIN-II $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ – voir le cours Protocoles & Réseaux de 2^e année

Un **pont** est en fait un nœud qui appartient aux deux réseaux couche 2 et dont le rôle consiste à réceptionner les trames d'un réseau pour les réémettre (ou pas...) vers l'autre réseau. Sa capacité à mémoriser les trames reçues est évidemment prépondérante dans le cas où la réémission serait ralentie par une surcharge de trafic sur le réseau destinataire.

2.4.1.5.2 Pont filtrant Le pont filtrant a pour fonction de ne transmettre les trames que sur le port qui mène en direction du destinataire. Le pont construit une table dynamique des adresses locales contenant l'adresse MAC source SA de chaque trame reçue et le numéro du port par lequel la trame est arrivée (algorithme de BARAN). Ainsi, on améliore le cloisonnement des trafics propres à chaque réseau au fur et à mesure que les tables d'adresses se remplissent. Pour lutter contre la saturation de ces tables et pour permettre les éventuels changements de port, les vieilles adresses finissent par être éliminées après un certain temps d'inactivité (*ageing*).

De plus, l'interconnexion par pont peut avoir lieu en mode local (*local bridge*) ou en mode distant (*remote bridges* – c'est-à-dire à travers une liaison privée ou publique entre les deux réseaux couche 2). La nature, la longueur et le débit de ces liaisons point à point peuvent varier considérablement, si bien que l'on est amené à délaisser le protocole 802.3 au profit de solutions particulières (comme des tunnels couche 2 via un réseau couche 3). Il est à signaler que le mouvement inverse se produit pour les infrastructures des opérateurs : de plus en plus, les protocoles spécifiques comme ATM sont abandonnés au profit de GBit Ethernet (avec VLAN et QoS et intégration MPLS).

2.4.1.6 Commutation Ethernet (switches)

Si les ponts permettent de cloisonner le trafic, il n'en reste pas moins que les réseaux ainsi séparés sont partagés par encore beaucoup de stations. L'expérience montre que ce type de réseau à contention (collisions) n'est exploitable que jusqu'à une charge soutenue de 30%.

Cette constatation a conduit à la notion de commutation Ethernet, qui est à la base des réseaux Ethernet switchés actuels. L'idée (de KALPANA) est d'isoler chaque station sur son propre segment de façon à lui permettre d'exploiter pleinement les 10 Mbps, 100 ou 1000Mbps du lien, si toutefois sa technologie (bus, disque...) en est capable.

Conceptuellement, un commutateur Ethernet (**switch**) n'est donc rien d'autre qu'un pont multiports filtrant rapide. Un traitement rapide est obtenu grâce à la technique du **forwarding cut-through**, qui consiste à émettre les trames reçues dès que le champ DA est analysé (l'adresse destination est en premier !) et donc à aiguiller les trames sur le bon port avant même de les avoir entièrement réceptionnées. L'inconvénient de relayer des trames corrompues faute de possibilité d'analyse de CRC est largement compensé par le faible retard (latence) induit par l'opération.

Pour les cas où le risque de collision subsiste sur le port d'entrée, il vaut mieux ne pas entamer la retransmission avant d'avoir attendu (>51.2 us) que le risque de collision soit écarté, et éviter ainsi de générer un fragment de trame sur le port de sortie (**forwarding fragment-free cut-through**). On peut encore relever que les équipements les plus intelligents sont capables de passer automatiquement du mode cut-through au mode classique store&forward (**fast forward**) en cas de problèmes excessifs de CRC.

Dans le cadre d'une architecture client/serveur, cette idée s'adapte parfaitement aux clients, mais elle ne résout que partiellement le goulet d'étranglement (**bottleneck**) vers le serveur qui peut se voir submergé par un débit cumulé des clients. Divers mécanismes permettent d'informer l'émetteur qu'il doit réduire, voire stopper, son émission (*back-pressure* en half-duplex, ou des trames 802.1q de contrôle de flux pour les versions actuelles d'Ethernet). D'autre part, la

majorité des commutateurs sont désormais dotés d'au moins un port *uplink* à 1 Gbps ou à 10Gbps, connecté au **backbone**.

Les switches sont notamment caractérisés par : leur débit interne total, que cela soit en $\frac{bit}{s}$ et en $\frac{paquet}{s}$, leur latence (dépendant également du mode d'exploitation de retransmission de trame), ainsi que les aspects de gestion (**managed switch**) : qualité de service, priorisation, VLAN, ...

2.4.1.7 Le routage couche 2 dans les réseaux Ethernet équipés de ponts

Le routage est une fonction de la couche 3 : toutefois, on peut parler également de *routage couche 2* pour l'optimisation d'**isolation** (lors de la **commutation** – *switching*) effectuée par les switches sur la base de leurs tables d'apprentissage (voir section 2.4.1.5.2 en page 44).

Un des buts de l'utilisation des ponts est de limiter le trafic dans les segments. Là où des répéteurs sont utilisés, tout le trafic est transmis sur tout le réseau. Les 100Mbps (ou 1000Mbps) se partagent entre toutes les stations car les répéteurs n'interprètent pas les informations de la couche MAC.

Les ponts eux ont accès à ces informations. Ils peuvent interpréter en particulier les adresses MAC de source et de destinations. Les réseaux Ethernet utilisent une méthode de routage appelée *routage transparent* (transparent routing en couche 2). Cette méthode fonctionne comme suit :

- les ponts lisent systématiquement les adresses source des trames reçues sur chaque port et enregistrent ces adresses avec l'identification du port. De cette façon le pont sait par quel port une station peut être atteinte (**isolation**)
- lorsqu'un pont reçoit une trame il cherche l'adresse destinataire correspondante dans sa table. S'il la trouve, il ne transmet la trame que sur le port qui permet d'atteindre cette station. S'il ne la trouve pas (ou s'il s'agit d'une adresse **broadcast**) il transmet la trame sur tous ses ports (sauf le port d'où la trame vient) (**flooding**, **inondation**).

Après un premier échange de trame, les adresses de deux stations qui communiquent sont enregistrées dans les ponts et les trames suivantes ne sont transmises que sur les segments nécessaires.

Il est évident que cette méthode ne peut fonctionner que si le réseau ne présente pas de boucle : la section 2.4.3.1 en page 46 montre le fonctionnement du protocole de l'arbre recouvrant, permettant de transformer un réseau LAN avec des boucles en un arbre, exploitable en Ethernet.

2.4.2 IEEE 802.2 LLC (informatif)

Logical Link Control (LLC) est une demi-couche haute de la couche 2 qui joue le rôle capital d'interface neutre et fédératrice entre les couches supérieures (3-7) et les couches inférieures (1-2). Il est à noter que les protocoles Ethernet IP n'utilisent *pas* cette sous-couche (dont le rôle, si nécessaire, est assuré par la couche 4 TCP) : elle est principalement utilisée pour des protocoles spécifiques de gestion réseau ou pour d'autres couche 2 qu'Ethernet (p.ex. **PPP**, basé sur **HDLC**).

Les services LLC peuvent être classés selon trois catégories fonctionnelles distinctes :

LLC1 service sans connexion et sans acquittement (LAN)

LLC2 service avec connexion (RNIS)

LLC3 service sans connexion et avec acquittement

Avec le service **LLC1**, la machine source envoie des trames à la machine destinataire sans recevoir d'acquittement(s) de la part de cette dernière. Aucune connexion n'est établie au préalable, ni libérée après l'envoi des données. Si une trame est perdue, aucun moyen n'est prévu pour y remédier. Ce type de service convient lorsque le taux d'erreurs est faible et lorsque le traitement des erreurs de transmission est prévu par les couches supérieures. Il est utilisé pour le trafic en temps réel et dans la majorité des réseaux locaux.

Le service **LLC3** est plus fiable que le précédent. Il n'existe pas de connexion, mais chaque trame envoyée est acquittée. L'émetteur sait donc pour chaque trame si elle a été correctement reçue. Si la trame n'est pas arrivée après un certain laps de temps, il peut l'envoyer de nouveau.

Avec le service **LLC2**, tout envoi de données doit être précédé de l'établissement de la connexion entre les machines source et destinataire. Un certain nombre de paramètres doivent donc être initialisés, en particulier des compteurs qui serviront à déterminer les trames qui ont été correctement transmises et celles qu'il faut retransmettre. Les primitives de service offertes par la couche 2 à la couche 3 sont les 4 primitives usuelles OSI (voir section 0.2.3.4 en page 12).

Du point de vue des trames 802.x, les champs LLC se situent dans les premiers octets du champ d'information avec la structure suivante (taille en octets) :

DSAP	SSAP	Contrôle LLC
1	1	1 (2)

Les deux premiers champs d'adresse identifient des points d'accès à un service de lien (LSAP) (Destination / Source Service Access Point) de la même manière que HDLC. L'assignation de ces adresses (en valeur décimale) est prévue par l'IEEE pour différents protocoles. D'autres entêtes existent ensuite suivant les protocoles.

2.4.3 Eléments avancés

2.4.3.1 Eviter les boucles en couche 2 : le spanning tree

La commutation (par inondation et cache géographique vue à la section 2.4.1.7 en page 45) ne fonctionne correctement que si le réseau ne comporte pas de boucle. Si des boucles sont présentes, le port par lequel on atteint une station n'est plus unique. C'est pourquoi l'utilisation du routage transparent est couplée à un algorithme exécuté en collaboration par les ponts qui a comme tâche de désactiver certains ports de certains ponts pour supprimer les boucles. Cet algorithme, qui consiste à produire un *arbre minimal recouvrant* tous les équipements du réseau bouclé, s'appelle le **spanning tree** (STP) et comporte les étapes suivantes :

- un des ponts est choisi comme racine (**root bridge**).
- chaque pont détermine quel est son port le plus proche de la racine (**root port**). La vitesse des liaisons est tenue en compte lors de cette évaluation. Ce port de chaque pont ne sera pas désactivé.
- on choisit dans chaque segment le port qui mène le plus rapidement à la racine (**designated port**). Un port ne peut être à la fois root port et designated port. Le designated port n'est pas désactivé.
- tous les ports qui ne sont ni root port ni designated port sont désactivés.

De cette façon le réseau ne comporte plus de boucle. L'algorithme est répété régulièrement. Si un port ou un segment tombe en panne, l'algorithme activera si possible d'autres ports. Le déroulement de l'algorithme nécessite l'échange de messages spécifiques entre les ponts.

Après le déroulement de l'algorithme le routage transparent (couche 2) peut être utilisé sans problème.

2.4.3.2 VLAN (Virtual LAN)

L'utilisation de switches pour limiter le trafic exige que les ressources et les utilisateurs soient attachés à la même couche 2 (par ex : les PC et les serveurs d'un groupe). Ceci est peu flexible et particulièrement gênant si des collaborateurs de groupes différents travaillent dans des locaux communs (double câblage) ou si les collaborateurs d'un groupe sont répartis sur plusieurs bâtiments, sans parler de problèmes lors de changements de place de travail ou de groupe. D'autre part la tendance aujourd'hui est de regrouper géographiquement les serveurs dans des locaux protégés, même si ces serveurs sont encore dédiés à un groupe. Un autre besoin est pour les applications de voix-sur-IP ou les équipements de **smart metering** ou **IoT** : on souhaite généralement les exploiter isolément. Enfin, la sécurisation peut nécessiter de mettre en place du **confinement**, séparant des réseaux administrativement.

La technologie des réseaux virtuels (**VLAN**) a été développée pour répondre à ces besoins. L'appartenance d'une station (PC, serveur, imprimante...) à un groupe n'est plus définie par le segment auquel elle est attachée mais par d'autres critères (port du switch, adresse MAC, subnet IP, login de l'utilisateur...). Le réseau doit alors pouvoir transmettre les trames d'un groupe uniquement jusqu'aux ports où des membres du groupe sont attachés. Deux problèmes doivent être résolus : qui détermine l'appartenance d'une trame à un groupe et comment transmettre cette appartenance afin de ne pas devoir la déterminer à chaque switch ?

Le premier problème se résout le plus souvent par l'appartenance géographique : le switch auquel la station est rattachée décide l'appartenance en fonction du port auquel la station est branché. Pour le second, le format des trames a été allongé (802.3ac) afin de transmettre l'appartenance à un réseau virtuel (802.1q) par une étiquette de VLAN (**tag**).

Ce format étendu ou *taggué* n'est en général utilisé que dans le **backbone** (épine dorsale, voir section 1.3.2.2 en page 30) du réseau, où plusieurs VLAN sont multiplexés [15] : c'est le mode **trunk**. L'utilisation de réseau virtuel est invisible pour les stations, qui sont en général connectés sur des ports en mode **access** (voir figure 2.4 en page 47). Des modes spécifiques comme les **VLAN privés** permettent d'implémenter un confinement entre stations, similaire au mode **client isolation** du WiFi.

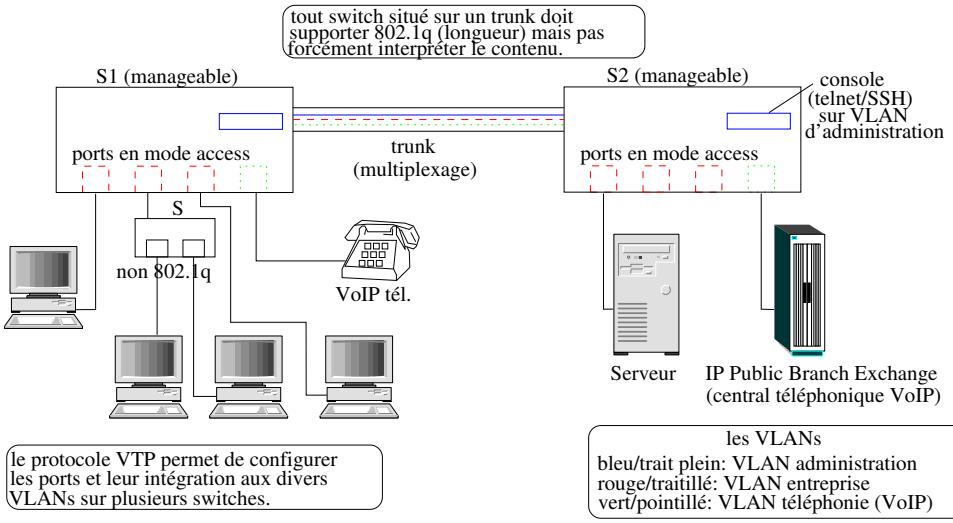


FIGURE 2.4 – Exemple de VLAN : backbone (trunk) et ports de stations (access)

En format étendu, la trame originale est donc rejetée 4 octets plus loin que d'habitude : en conséquence, la longueur totale maximale d'une trame (**MTU**, *Maximum Transmission Unit*)

15. chaque trame doit alors être **tag**guée, sauf éventuellement pour le VLAN par défaut : le **native** VLAN

couche 2 est portée de 1518 à 1522 bytes, ce qui nécessite une modification de la couche MAC – ce qui peut poser des problèmes à de très anciens équipements s'ils sont utilisés sur des trunks.

Le champ Type contient une valeur particulière (0x8100) indiquant la présence de **tag** de VLAN **802.1q**, encodés dans le format étendu **802.3ac**. Le champ Tag Control contient des informations concernant la priorité de la trame (3 bits, **802.1p**) et l'identification du réseau virtuel (12 bits). Le champ T/L correspond à la valeur originale du type ou de la longueur de la trame avant multiplexage :

PA + SFD 62 + 2	DA 48	SA 48	**Type 802.1q 16**	**Tag control 16**	T/L 16	INFO (+PAD) 368-12000	CRC 32

2.4.3.3 Agrégation de liens couche 2

Alors que les VLANs permettent le multiplexage de plusieurs flux sur un seul canal, mutualisant le débit total, le but d'une agrégation de liens en couche 2 est d'augmenter le débit disponible : par exemple, en utilisant deux liens cuivre à 8 fils UTP 6, il est possible de doubler le débit Ethernet de 10 GBit/s à 20 GBit/s (**load balancing**). Lorsque les distances l'exigent, on utilisera de la fibre optique dans laquelle on peut déployer du **multiplexe de couleurs** (**CWDM**) permettant de subdiviser l'espace fréquentiel (couleur) d'une fibre de manière très économique en 16 canaux (à 1 GBit/s maximum) ou 8 (jusqu'à 10 GBit/s) permettant un débit total maximum jusqu'à 80 GBit/s sur une distance d'un peu plus de 100 km.

EtherChannel est un protocole spécifique à Cisco qui permet d'utiliser plusieurs liens Ethernet pour échanger du trafic entre deux switches. Son principe est d'utiliser un hachage qui détermine par quelle port passera une trame donnée, par exemple basé sur l'adresse MAC, l'adresse IP, le numéro de port, etc. Il est possible de combiner la performance avec la **haute fiabilité** : soit en **mode dégradé** (moins de liens utilisés), en mode **fail-over** avec des liens de réserve permettant de maintenir la performance ou encore avec le protocole **STP** (spanning-tree protocol, voir 2.4.3.1 en page 46). Une version standardisée de ce protocole existe sous la dénomination IEEE **802.1ax** (**LACP** – *Link Aggregation Control Protocol*).

2.4.3.4 Wake-on-LAN

Le *Wake-on-Lan* (**WoL**) permet de réveiller des machines en veille grâce à une trame Ethernet *magique* traitée par la carte réseau en demi-sommeil (très basse consommation) reconnaissant des répétitions spéciales de son adresse MAC. La portée originale du service était en général limitée à la portée de couche 2 (équivalent à un sous-réseau en IP), mais des relais peuvent être installés, et dans certains cas (sauf firewall), des paquets UDP/IP peuvent être envoyés via Internet.

2.4.3.5 Mesh

La norme IEEE **802.1aq** (*Shortest Path Bridging*) remplacera à terme à la fois le spanning tree et l'agrégation de liens et permettra la redondance et la performance au sein d'un **mesh** (réseau maillé), complètement automatiquement : grâce à elle, Ethernet se maintiendra comme protocole universel de couche 2, que cela soit pour les réseaux locaux d'entreprise ou les MAN d'opérateurs. Cette norme fait partie du concept émergent de *Software Defined Networking* (**SDN**) : l'idée d'appliquer la virtualisation au réseau lui-même (voir section 0.3.7 en page 20).

Chapitre 3

Couche réseau (3)

7	Application
6	Présentation
5	Session
4	Transport
3	**Réseau**
2	Liaison
1	Physique

Sommaire

Le but de ce chapitre est de présenter le rôle de la couche réseau (en anglais *network layer*), la couche qui réunit les nombreuses liaisons point à point ou point à multipoint de couche 2 dans un réseau cohérent de bout en bout. Nous nous concentrerons sur les incontournables exemples d'Internet et du protocole IP, vu son ubiquité et sa popularité.

3.1 Internet et IP

3.1.1 Introduction

3.1.1.1 Définitions

Un **internet** est une fédération de réseaux interconnectés. Le réseau **Internet** (avec une majuscule) est l'internet planétaire à protocole IP. Cette multitude de réseaux interconnectés représente un maillage planétaire d'appareils de communication et d'ordinateurs adressables à travers une pile de protocoles communs standardisés que l'on connaît sous le nom de TCP/IP [1] et que l'on peut modéliser au sein du modèle OSI (voir section 0.2.3 en page 8), souvent dans un modèle simplifié à 5 couches (voir figure 9 en page 18).

3.1.1.2 Historique

Internet est un réseau prévu dès le départ pour le transport des données informatiques, conçu dans le cadre du projet DARPA (*Defense Advanced Research Project Agency*) par la communauté scientifique, de recherche et les utilisateurs universitaires. Ses premiers standards datent de 1969. Ses ancêtres sont les projets de recherche dans le domaine des réseaux à paquets dans les années 50 et 60, notamment en France. Sa finalisation date du milieu des années 80, encore que des améliorations sont proposées et apportées de manière continue via les **RFC** (*Request for Comment*) et documents de standards **STD** et de bonnes pratiques **BCP**. Il a été développé pour être utilisé dans les milieux universitaires puis s'est ouvert au monde entier dans les années 1990. La bonne intégration des protocoles d'Internet dans **UNIX** a beaucoup contribué à l'expansion originale d'Internet (1980-1990).

Internet et ses protocoles sont actuellement incontournables. Ils sont supportés par tous les types d'ordinateurs et tous les systèmes d'exploitation, du serveur de calcul distribué au téléphone mobile (**smartphone**).

IP version 4 a pris une importance énorme, écrasant tous ses concurrents potentiels. Cependant, certains de ses concepts de base ne sont plus toujours adaptés aux dimensions des réseaux actuels, aux volumes des données transférées ni aux usages qu'on souhaite en faire (transport de la voix, sécurité des données...). Pour cette raison, en plus des améliorations constantes, une nouvelle version du protocole a été proposée dans les années 90 et est en cours de mise en production (**IPv6**, voir section 3.3.2 en page 76). De plus, les opérateurs télécoms, dont les besoins sont particuliers (facturation, roaming, chiffrement permettant des écoutes par les autorités nationales, **QoS**, etc) sont entrain de définir des efforts d'intégration entre leurs réseaux classiques et les protocoles Internet, au travers des *Next Generation Networks* (**NGN**), voir section 3.3.4 en page 78.

1. abus de langage lié à des raisons historiques de confusion entre couche 4 (TCP ou UDP) et couche 3 (IP)

3.1.1.3 Approche comparée OSI/IP

Internet étant un réseau, le protocole essentiel est celui de la couche 3 (IP). Alors que dans les couches plus hautes et plus basses différents protocoles sont utilisés, tout Internet n'utilise qu'un protocole pour la couche 3. Il a été choisi de rendre cette couche aussi simple que possible [2], en limitant le travail effectué par les nœuds [3] intermédiaires de couche 3 : les **routeurs**.

C'est une approche très différente de celle choisie par l'ITU ou l'ISO. Eux ont voulu que le fournisseur des services de télécommunication puisse offrir des services fiables pour ses clients. En conséquence les protocoles de la couche 3 dans ce modèle doivent offrir un transport fiable des données et cette couche devient complexe. Les concepteurs d'Internet n'étant pas des fournisseurs de services, ils ont admis que les réseaux n'étaient jamais complètement fiables et que l'utilisateur devait se prémunir lui-même contre les problèmes du réseau. C'est pourquoi la couche IP fonctionne sans connexion et sans confirmation, ni qualité de service exploitable sur Internet.

L'approche qui de fait a gagné est celle d'Internet : mais après un remplacement de certains protocoles des opérateurs par les protocoles de la pile TCP/IP, des intégrations entre ces deux approches sont de plus en plus mises en oeuvre : que cela soit en couche 2 avec les grands réseaux **core** des opérateurs fonctionnant sur **MPLS** (voir section 3.2.3.4 en page 71) ou à travers tout le modèle OSI via les **NGN** et une intégration de services provenant largement du monde IP (voir section 3.3.4 en page 78).

3.1.2 Adressage

3.1.2.1 Adressage et classes

Le protocole Internet utilise 4 octets (32 bits) pour identifier un ordinateur hôte ainsi que le réseau auquel il est rattaché. Les adresses IP (RFC-791) se présentaient *initialement* sous la forme des cinq catégories ou **classes** suivantes (les bits **MSB**s, *Most Significant Bits*, en commençant tout à gauche dans le codage binaire, définissent la classe) :

classe	MSBs	1er octet	bits réseau	bits host
A	0	0-127	7	24
B	10	128-191	14	16
C	110	192-223	21	8
D	1110	224-239	adresse multicast (28 bits), voir section 3.1.4.3 en page 57	
E	1111	240-255	réservé	

Sauf pour les classes D et E ci-dessus, le concept de classe est désormais *obsolète* et remplacé par le concept de sous-réseau (**subnetting** – basé sur un *masque de sous-réseau variable*, voir section 3.1.3 en page 52).

Afin de simplifier la présentation, les adresses IP sont habituellement représentées sous la forme de 4 nombres décimaux [4] séparés par des points et représentant les 4 octets. L'adresse binaire 10000000 00000011 00001001 00000001 est par exemple écrite 128.3.9.1 et représente l'hôte (la machine) 9.1 du réseau 128.3 (classe B). Le premier octet indique donc la classe du réseau considéré (voir la 3^e colonne du tableau ci-dessus).

2. cette tendance est encore plus visible avec **IPv6**.
3. la terminologie IP désigne par **nodes** (**nœuds**) à la fois les routeurs et les terminaux (voir page 3).
4. sans représenter de zéros initiaux : 128.003.009.001 n'est pas une adresse généralement correcte même si on la trouve parfois dans des exemples ou des GUI de configuration, car le préfixe 0 dénote usuellement la base 8 dans les systèmes POSIX.

3.1.2.2 Allocation dynamique des adresses

Plutôt que de configurer manuellement une ou plusieurs adresses sur chaque interface de chaque équipement, un protocole a été défini : le **DHCP** (*Dynamic Host Configuration Protocol*). Il permet, par découverte (via **broadcast** IP/Ethernet), d'obtenir les informations de configuration (adresse IP, masque de sous-réseau, routeur par défaut, serveurs divers : DNS (noms), NTP (temps), démarrage, etc). Evolution du protocole **BOOTP**, il nécessite un serveur dans le sous-réseau vu l'usage de broadcasts limités (voir section 3.1.4.2 en page 57) – ou au moins un relais DHCP.

Il est d'usage de protéger les réseaux d'entreprise de serveurs DHCP mal configurés en les bloquant sur les ports non dédiés à un serveur DHCP (p.ex. Cisco **DHCP snooping**).

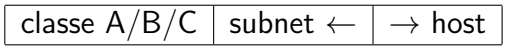
L'utilisation d'adresses dynamiques au sein d'un *pool* se justifie pleinement pour les équipements clients. Il est aussi possible de faire retourner par le serveur DHCP des adresses IP associées à une adresse MAC particulière (p.ex. pour des équipements comme des imprimantes).

En l'absence d'un serveur DHCP et de configuration manuelle, des adresses de portée lien (voir section 3.1.3.7 en page 56) peuvent être autoconfigurées. D'autres protocoles peuvent gérer l'autoconfiguration pour une couche 2 différente d'Ethernet. Citons l'autoconfiguration de **PPP** (xDSL ou VPN par exemple).

3.1.3 Subdivisions de réseaux

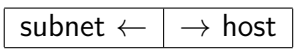
3.1.3.1 Sous-réseaux

Initialement, les classes A à C définissaient les réseaux disponibles de manière fixe. Ensuite, il y a été possible de subdiviser de manière variable [5] la partie hôte des anciens réseaux classes A à C (obsolètes) en une partie sous-réseau (subnet) et une partie hôte restreinte, pour plus de flexibilité.



Pour ce faire, on recourt à un masque de sous-réseau (**subnet mask** ou **netmask**) qui permet de distinguer la partie réseau (bit à 1) de la partie hôte (bit à 0) en fonction des besoins, de cas en cas et non plus de manière fixe comme avec les classes. Le masque 255.255.255.0 convient ainsi à un réseau de classe C sans aucun sous-réseau, ou encore à un réseau de classe B comprenant jusqu'à 256 sous-réseaux. Par contre, le masque 255.255.255.192 autorise 4 sous-réseaux d'au maximum 64 stations pour la classe C. On peut garder en tête qu'une adresse IP est combinée avec le subnet mask par un ET logique, avec toutes les possibilités de bits fixés entre et y compris 32 (une adresse) et 0 (tout l'espace d'adressage [6]).

Un sous-réseau est alors à exprimer en donnant son adresse et son **netmask**, par exemple : 157.26.77.0/255.255.255.0 (sous-réseau de 256 adresses). C'est la méthode utilisée aujourd'hui pour définir des sous-réseaux (les classes étant maintenant désuètes).



Rappelons que toutes les adresses ne sont pas utilisables : la 1ère adresse d'un sous-réseau est réservée à l'adresse de ce sous-réseau, la dernière à l'adresse **broadcast** et il faut encore, en

5. Cisco appelle cela *Variable Length Subnet Mask*, ou VLSM, voir aussi la section 3.1.3.2 en page 53
6. voir par exemple le concept de route par défaut à la section 3.2.2.2 en page 68.

règle générale, prévoir une adresse pour l'interface d'un routeur servant à atteindre l'extérieur (**route par défaut**, passerelle[7]).

3.1.3.2 Adressage CIDR

Une façon simplifiée – et celle que l'on devrait utiliser – d'exprimer une adresse de sous-réseau est de remplacer le netmask par le nombre de bits fixes dans l'adresse (adresse-sous réseau).

Par exemple, le sous-réseau 157.26.77.0/255.255.255.0 peut s'exprimer 157.26.77.0/24 en notation **CIDR**, *Classless Inter-Domain Routing*.

Pour convertir en (sub)netmask, si l'on connaît le nombre de bits fixés (p.ex. /27), il est facile de déterminer que les 3 premiers octets du netmask seront fixés à 1 (car 3 fois 8 vaut 24), et que le dernier octet aura ses 3 bits MSB (27 - 24) fixés à 1 (soit 128 + 64 + 32), le reste à zéro. Donc, le netmask d'un /27 vaudra 255.255.255.224. Une autre manière d'obtenir le résultat est d'observer qu'un /27 est un sous-réseau avec 32 - 27 soit 5 bits libres ou encore $2^5 = 32$ adresses possibles, et que 256 - 32 vaut bien 224.

De plus en plus d'équipements peuvent être configurés directement en CIDR : c'est la seule forme standard avec IPv6. D'anciennes interfaces doivent encore être configurées avec l'adresse de sous-réseau et le netmask.

3.1.3.3 Exemples de sous-réseaux dans le contexte CIDR

nombre d'adresses	bits fixes (à 1) et bits libres (à 0)	(sub)netmask	CIDR partiel (bits fixes)
1	11111111 11111111 11111111 11111111	255.255.255.255	/32
2	11111111 11111111 11111111 11111110	255.255.255.254	/31
4	11111111 11111111 11111111 11111100	255.255.255.252	/30
8	11111111 11111111 11111111 11111000	255.255.255.248	/29
16	11111111 11111111 11111111 11110000	255.255.255.240	/28
32	11111111 11111111 11111111 11100000	255.255.255.224	/27
64	11111111 11111111 11111111 11000000	255.255.255.192	/26
128	11111111 11111111 11111111 10000000	255.255.255.128	/25
256	11111111 11111111 11111111 00000000	255.255.255.0	/24
512	11111111 11111111 11111110 00000000	255.255.254.0	/23
1024	11111111 11111111 11111100 00000000	255.255.252.0	/22
2048	11111111 11111111 11111000 00000000	255.255.248.0	/21

. . . . . .

Les sous-réseaux /32 et /31 sont dégénérés : /32 ne désigne qu'une adresse (p.ex. dans des règles de firewall). Et /31 ne contient que deux adresses pour une liaison point à point (par exemple pour deux routeurs, mais sans adresses de sous-réseau ni de broadcast) : voir RFC-3021.

Indications :

— si l'on a besoin d'un sous-réseau de n adresses, alors il faut $\lceil \log_2 n \rceil$ bits libres (à arrondir à l'entier supérieur), et le nombre de bits fixés est alors $32 - \lceil \log_2 n \rceil$

— le netmask est (sur le dernier octet) : $256 - n$ (n arrondi à la puissance de deux supérieure)

7. attention, le concept de passerelle (*gateway*) correspond dans le modèle OSI à une conversion de protocole de couche 7 – ici on l'utilise dans le sens usuel Internet.

Exemple d'allocation d'un sous-réseau On veut un sous-réseau contenant deux adresses de machines. Il faut donc 5 adresses (première : adresse sous-réseau, dernière : adresse broadcast (envoi à tous), plus l'adresse de l'interface du routeur). Au minimum, on doit donc allouer un sous-réseau à 8 adresses. Le netmask est donc 255.255.255.248. Exemples de sous-réseaux possibles au sein de 193.72.186.0/24, en notation CIDR : 193.72.186.0/29, 193.72.186.8/29, 193.72.186.16/29, ..., 193.72.186.248/29.

Autre exemple : pour 64 adresses au total : netmask 255.255.255.192, exemples 80.83.54.0/26, 80.83.54.64/26, 80.83.54.128/26, 80.83.54.192/26.

Appartenance à un sous-réseau Pour vérifier si deux adresses de machines sont dans le même sous-réseau, appliquer le netmask. Par exemple, les adresses 59.20.251.140 et 59.17.42.2, sont-elles les deux dans le sous-réseau 59.16.0.0/13 ?

Les bits fixes sont $13 = 8 + 5$, il y a donc 3 bits libres dans le deuxième octet et le netmask vaut 255.248.0.0. On calcule 59.20.251.140 AND 255.248.0.0, qui vaut 59.16.0.0, et 59.17.42.24 AND 255.248.0.0, qui vaut aussi 59.16.0.0, ils sont dans le même sous réseau 59.16.0.0/13 : la plage décrite par le sous-réseau est d'ailleurs 59.16.0.0 à 59.23.255.255 (3 bits de libres dans le deuxième octet, fait 8 possibilités : de 16 à 23).

Subdivisions d'un sous-réseau Un sous-réseau ne peut débuter que sur une adresse multiple du nombre d'adresses du sous-réseau. Dans l'exemple ci-dessus, 59.20.251.140 est dans le sous-réseau 59.16.0.0/13 car un /13 désigne toujours une adresse de sous-réseau dont le 2e octet est un multiple de 8 (3 bits de libres dans le 2e octet) et 0, 16, 24 ... sont des multiples de 8 possibles dont seul 16 à 23 contient la valeur 20. C'est aussi la raison pourquoi, si l'on veut subdiviser un sous-réseau, il vaut mieux le faire par sous-réseaux de *taille décroissante*.

3.1.3.4 Supernetting (agrégation de sous-réseaux)

Dans la mesure où un fournisseur vous donne accès à deux sous-réseaux contigus (au sens qu'ils ne diffèrent que par le bit fixé le plus à droite), il est possible de les réunir par supernetting en un seul sous-réseau CIDR. Il est bien sûr possible de généraliser à plus de deux sous-réseaux contigus, s'il s'agit d'une puissance de 2 et que les bits au départ fixés définissent entièrement [8] la plage indiquée. Graphiquement, on déplace la frontière entre bits fixés et libre vers la gauche.

Par exemple : les deux sous-réseaux 157.26.77.0/24 et 157.26.76.0/24 peuvent se réduire à 157.26.76.0/23. Ou encore, les quatre sous-réseaux 157.26.76.0/24, 157.26.77.0/24, 157.26.78.0/24 et 157.26.79.0/24 peuvent se réduire [9] à 157.26.76.0/22.

Contre-exemple : 193.72.185.0/24 et 193.72.186.0/24 ne peuvent former un /25 : mais on pourrait fusionner 193.72.184.0/24, 193.72.185.0/24, 193.72.186.0/24 et 193.72.187.0/24 via un /22, si ces 4 sous-réseauyx vous appartiennent.

L'avantage principal est la réduction du nombre d'entrées nécessaires dans les tables de routage et donc l'économie de mémoire dans les routeurs, par l'agrégation de routes que cela produit.

3.1.3.5 Allocations d'adresses publiques

Ce sont les registres Internet régionaux (**RIR**s, p.ex. **RIPE** en Europe) qui fournissent des plages d'adresses : la plupart du temps à des **LIR**s (*Local Internet Registry*), qui les fournissent aux

8. **WHOIS** confirme que le sous-réseau fusionné vous appartient entièrement
9. car 76 est un multiple de 4

systèmes autonomes (voir 3.2.3.3.2 en page 70). Pour les utilisateurs finaux ou les entreprises sans besoins spécifiques, ce sont les fournisseurs d'accès Internet (**FAI**) qui s'en chargent, en allouant dans leurs propres plages. Les informations sont centralisées dans des bases de données (**WHOIS**) et utilisées pour la gestion du routage.

Aujourd'hui et dans la plupart des cas, les adresses mises à disposition par les fournisseurs aux utilisateurs finaux sont dynamiques (peuvent changer au cours du temps au sein d'un ou plusieurs *pool*) et donc ne sont pas adéquates [10] à l'exploitation de services classiques Internet. Les versions *pro* des accès Internet permettent de louer une ou plusieurs adresses, voire des plages aux FAI dans la plupart des cas.

L'actuel réseau HE-Arc est bâti, comme le réseau pédagogique des Ecoles neuchâteloises (RPN, Université), sur l'ancienne classe B publique 157.26.0.0/16 – allouée à l'époque par l'Ecole d'ingénieurs du Locle – dans un système autonome (**AS**, voir section 3.2.3.3.2 en page 70) séparé. Tous les équipements de l'Ecole ont une adresse IP publique routable, et s'il n'y avait un **firewall**, elles seraient accessibles de tout Internet sans restriction.

Cette organisation est aujourd'hui relativement rare : on doit constater qu'il est quasiment impossible aujourd'hui d'obtenir des plages d'adresses publiques de cette taille : pour les nouveaux déploiements, on obtiendrait vraisemblablement des sous-réseaux plus petits en fonction des besoins (des /24, voire des sous-réseaux encore plus petits), vraisemblablement disjoints, ou, plus vraisemblablement, on passerait à des adresses IP privées (voir section 3.1.3.6 en page 55) et à une translation d'adresse (**NAT**, voir section 3.2.6) et/ou un proxy-application pour accéder Internet.

Aujourd'hui et la plupart du temps, les adresses qui sont fournies à une entreprise sont allouées dans la plage d'un fournisseur d'accès à Internet : ces adresses ne sont donc pas propriété de l'entreprise et sont dépendantes du fournisseur (**plage PA**). Dans des configurations particulières (p.ex. multi-homing voir section 3.2.4 en page 71), des plages d'adresses portables ou indépendantes d'un fournisseur sont – de plus en plus difficilement [11] – allouables (**plage PI**).

Il est à prévoir que le besoin en migration à **IPv6** deviendra de plus en plus important au fur et à mesure des nouveaux besoins (en particulier dans les pays émergeants n'ayant pas disposé de larges allocations initiales de l'espace d'adressage 32 bits IPv4).

3.1.3.6 Adresses privées

Lorsqu'on ne dispose pas des adresses publiques nécessaires ou s'il n'y a pas de volonté de connecter un réseau à Internet directement, on peut utiliser des plages d'adresses particulières, nommées privées et qui ne sont pas routables sur Internet, mais uniquement à l'intérieur d'un réseau particulier.

préfixe CIDR	taille totale	plage d'adresses	nombre d'adresses
10.0.0.0/8	24 bits	10.0.0.0 - 10.255.255.255	16'777'216
172.16.0.0/12	20 bits	172.16.0.0 - 172.31.255.255	1'048'576
192.168.0.0/16	16 bits	192.168.0.0 - 192.168.255.255	65'536

Ces réseaux peuvent être subdivisés totalement comme on le désire. Par exemple, on peut diviser 192.168.0.0/16 en 256 sous-réseaux 192.168.x.0/24, ou on peut découper d'une autre manière

10. il est possible d'utiliser des enregistrements DNS à courte durée de vie, mais c'est risqué – voir par exemple le service http://www.dyndns.org/ et la section 3.1.6.3.6 en page 67.

11. RIPE annonce la procédure d'allocation finale le 14 septembre 2012, dans laquelle chaque membre LIR obtient un /22 du dernier /8 de RIPE https://www.ripe.net/internet-coordination/ipv4-exhaustion, et, le 25 novembre 2019, il n'y a plus d'adresses libres ; la situation est similaire dans les autres régions

(un seul sous réseau 192.168.0.0/16, deux sous-réseaux 192.168.0.0/17 et 192.168.128.0/17...). C'est effectivement le réseau privé 10.0.0.0/8 qui offre le plus de flexibilité.

Ces adresses privées peuvent utilisées dans le but d'accéder Internet via de la translation d'adresses (**NAT**, **PAT**, voir section 3.2.6 en page 73) sur un **firewall**, ou via un **proxy**.

3.1.3.7 Autres adresses réservées ou particulières (RFC-5735)

En plus des adresses privées, des anciennes classes E (réservée) et D (multicast), il y a d'autres plages affectées à des utilisations particulières, par exemple 127.0.0.0/8 (utilisée notamment pour l'adresse de bouclage **127.0.0.1** qui représentente l'équipement lui-même), ou des plages utilisables pour la documentation ou les tests.

Citons encore le sous-réseau de portée lien (**link-local**) RFC-3927 **169.254.0.0/16** qui est utilisé en cas d'absence de configuration[12] manuelle ou dynamique (**DHCP**) : le protocole **IPv4LL**[13] (adressage IPv4 en portée de couche liaison, excluant tout routage extérieur), implémenté sous UNIX par le daemon **avahi** et sous MacOS par **Bonjour**, correspond fonctionnellement aux adresses **link-local** (portée lien) d'IPv6 **fe80**::**/10%lien**[14].

3.1.4 Modes d'envoi (unicast, diffusion, multicast et anycast)

3.1.4.1 Introduction

Le mode usuel d'envoi d'un datagramme (ou d'une trame en couche 2) est le mode **unicast**, soit d'une machine à une machine. Tous les modes possibles[15] sont décrits en détail ci-après :

mode	description
unicast	envoi à une station particulière
broadcast	envoi à toutes les stations d'un sous-réseau ou d'une couche 2 spécifique, voir section 3.1.4.2 en page 57 (n'existe pas en IPv6, mais est émulé sous forme d'un multicast au groupe de toutes les machines)
multicast	envoi à toutes les stations qui se sont abonnées au groupe multicast considéré (via protocoles PIM ou IGMP, voir section 3.1.4.3 en page 57)
anycast	envoi à une machine parmi un groupe (IPv6 seulement), utile pour les services redondants (haute disponibilité)

Il est à noter que le multicast n'est implémenté qu'imparfaitement en dehors d'un ensemble de réseaux coopérants étroitement. Il y a espoir, pour le moment non réalisé, qu'IPv6 ou les NGNs puissent généraliser le multicast à travers Internet : cette amélioration aurait un impact non négligeable sur le coût de la multi-diffusion, par exemple pour les fournisseurs de contenu (radio/TV Internet, Youtube...) – pour plus de détail consulter la section 3.1.4.3 en page 57.

Enfin, la notion d'adressage **anycast**, soit l'envoi à un destinataire parmi N d'un groupe, n'existe pas en IPv4. Il est possible de l'émuler par diverses techniques au sein d'un réseau contrôlé, voire par des annonces de routage globales spécifiques (**BGP**). L'usage principal de

12. **Zeroconf** est la spécification IETF d'autoconfiguration d'adresses, de noms ou de service
13. Microsoft quant à lui a nommé ce protocole **APIPA**
14. en v4, il manque l'identificateur de portée lien `%lien` : le sous-réseau doit être unique sur le *host* et pas seulement sur le *lien* comme en v6
15. unicast, broadcast et multicast sont aussi implémentés en couche 2 Ethernet

l'anycast est pour la redondance ou la performance. Un candidat évident sont les **root servers** (serveurs racines du DNS).

3.1.4.2 Broadcasting (diffusion)

Le principe du broadcasting est d'envoyer un datagramme à toutes les stations (que cela soit toutes les stations reliées à une même couche 2 Ethernet, toutes les stations d'un sous-réseau particulier, ou tous les sous-réseaux).

Les implémentations de piles IP modernes ne répondent pas par défaut aux broadcasts **ping** pour éviter l'**attaque par tempête** (**broadcast storm**).

Ci-après, vous trouverez les différents types d'adresses de diffusion (**broadcast**), en supposant le sous-réseau 157.26.77.0/24 – situé dans une ancienne classe B 157.26.0.0/16 – en ne traitant que la couche 3 [16].

type	format	exemple	description
dirigé	adresse du réseau indiquée telle quelle, la partie host a ses bits à 1	157.26.255.255	toutes les stations du réseau couche 3 spécifié ; pas forcément fonctionnel : peut être filtré
limité	tous les bits à 1	255.255.255.255	toutes les stations de *cette* couche 2 quel que soit leur sous-réseau.
vers un subnet	partie fixée du sous-réseau indiquée telle quelle, partie libre a ses bits à 1	157.26.77.255	toutes les stations du sous-réseau couche 3 spécifié ; peut être filtré.

3.1.4.3 Multicasting

Certaines applications multimédia, mais ce ne sont pas les seules, consistent souvent à diffuser à un nombre de participants les mêmes données audio ou vidéo. Créer autant de flux qu'il y a de participants (**streaming unicast**) crée un grand trafic réseau et le serveur à l'origine de la *source* de données devient le goulet d'étranglement : c'est pourtant ce qui se passe par exemple avec le streaming (Youtube, radios MP3...) actuellement.

L'idée de base de la multi-diffusion ou multicast est de charger le réseau des tâches d'acheminement des flux multiples. Cela signifie que ce sont les switches (en couche 2, qui filtrent) et les routeurs (en couche 3) qui s'occupent de la génération de flux multiples depuis un flux unique original, ce qui crée un véritable **streaming multicast** sous la forme d'un arbre dont la racine est le serveur d'origine.

En IPv4 (RFC-988), l'implémentation définit les concepts de *groupe* multicast, et d'*abonnement-désabonnement* à un groupe multicast. Les fonctions d'abonnement et de désabonnement sont fournies par le réseau sans que la source ne soit consultée, par exemple par le protocole **IGMP** (*Internet Group Management Protocol*) et/ou **PIM**. Cela signifie que d'éventuelles fonctions de sécurité (autorisation d'accès à un flux) doivent être implémentées par l'application à un autre niveau. De plus, le fonctionnement même du multicast rend l'utilisation de TCP impossible : seul UDP sera utilisé.

16. en couche 2, un broadcast atteint toujours toutes les machines de la même couche 2, car l'adresse MAC destination avec tous les bits à 1 est utilisée ; ces broadcasts couche 2 ne traversent pas les routeurs

Au niveau technique, les adresses de classe D sont utilisées (les 28 bits de poids faible forment l'adresse du groupe multicast). Certaines adresses sont réservées pour des protocoles particuliers (p.ex. **OSPF**, voir 3.2.3.2.2 en page 69).

Ce sont les routeurs qui convertissent les adresses multicast couche 3 en adresses multicast couche 2 qui sont reçues par les cartes réseaux des équipements abonnés au groupe. La conversion RFC-1112 consiste à placer les 23 bits inférieurs du numéro de groupe multicast dans l'adresse MAC destination, préfixée de `01:00:5e/25`, ce qui permet tout de même aux cartes réseau d'ignorer les groupes auxquels aucune application n'est abonnée, s'ils différent dans les 23 bits inférieurs.

Les switches (couche 2) répètent simplement ces trames à tous les équipements connectés (sauf si l'**IGMP snooping** est utilisé).

Le multicast est cependant peu utilisé, en particulier hors de réseaux spécifiques : même si IPv6 rend le multicast obligatoire, car lié aux fonctions de base du réseau (notamment pour le remplacement d'**ARP**, le protocole **NDP**, *Neighbour Discovery Protocol*), il n'y a toujours pas de gestion administrative universelle des groupes multicast par l'utilisateur.

Deux alternatives efficaces au multicast sur Internet existent, pour remédier au fait qu'il n'est pas supporté sur l'Internet global : une première consiste à s'abonner à des réseaux de distribution de contenu (**CDN**, *Content Distribution Network*) ; la seconde est le **P2P** (*Peer To Peer*) : plus de clients consomment de données, plus ils peuvent également en offrir à des tiers, garantissant la mise à l'échelle du protocole ; appliqué à un système de fichiers comme **ipfs**, c'est une piste intéressante à suivre (voir section 7.2.7 en page 124).

3.1.5 Entête IP

3.1.5.1 Format général

En plus de l'adressage, la couche Internet (OSI couche 3) définit un format d'entête spécifique, encapsulé après les entêtes couche 2 et avant les enqueues (postêtes, trailers) de la couche 2 : avec Ethernet, le CRC-32 serait situé tout à la fin (voir couche 2 de la figure 5 en page 14, champ DT) mais il n'est en général pas visible dans les captures.

Tous les entêtes sont envoyées en **big endian**, comme beaucoup de protocoles réseaux : les processeurs Intel x86 doivent donc retraiter les données. Les nombres entre parenthèses sont le nombre de bits de chaque champ et chaque ligne représente 32 bits.

<table>
<tr><td>Version (4)</td><td>IHL-Header Len (4)</td><td>Type Of Service (8)</td><td colspan="2">Total Length (16)</td></tr>
<tr><td colspan="3">Identifier (16)</td><td>Flags (3)</td><td>Fragment Offset (13)</td></tr>
<tr><td colspan="2">Time To Live (8)</td><td>Protocol (8)</td><td colspan="2">Header Checksum (16)</td></tr>
<tr><td colspan="5">Source IP Address (32)</td></tr>
<tr><td colspan="5">Destination IP Address (32)</td></tr>
<tr><td colspan="5">Options et bourrage (si existe)</td></tr>
</table>

<table>
<tr><td>Données des couches supérieures</td></tr>
</table>

FIGURE 3.1 – Entête IP (version 4)

Le protocole IP est actuellement en version 4. L'entête a en principe une longueur de 5 mots de 32 bits. Cela signifie que le premier octet de l'entête IP vaudra en règle générale 0x45

(hexadécimal), sauf s'il y a des mots d'options (avec **bourrage**[17] éventuel pour atteindre un nombre entier de mots de 32 bits d'options).

3.1.5.2 Champ Flags

Ce champ de 3 bit (plus de détail à la section Fragmentation 3.1.5.4 en page 59) se décompose comme suit :

réservé	DF (don't fragment)	MF (more fragments)

3.1.5.3 Champ TOS (Type of Service)

Le champ **DSCP** (*Differential Services Code Point*) est prévu pour la gestion de la qualité de service de type **DIFFSERV**. Sa définition a évolué : les bits **TOS** (optimisation du débit, du délai, de la fiabilité ou du coût monétaire) sont activés par les équipements au besoin et les routeurs peuvent en tenir compte, et le champ *Precedence* peut être utilisé pour définir des classes de priorité ; ce champ n'est pas supporté globalement sur Internet mais uniquement au sein d'un réseau administratif cohérent.

Par exemple, le réseau **core** d'un grand opérateur suisse utilise les bits TOS pour classifier le trafic dans la mesure où le réseau n'est pas surchargé ou que la classe **DSCP** désigne du trafic prioritaire. Il est d'usage d'effacer les bits TOS à l'entrée d'un réseau – ou du moins les bits de Precedence, par exemple selon que le client a effectivement payé ou non pour le service demandé.

7	6	5	4	3	2	1	0
ECN, RFC-3168		**DSCP**, RFC-2474					
		R (fiabilité)	T (débit)	D (délai)	Precedence (classe de priorité)		

(par exemple)

Le sous-champ **ECN** est une amélioration utile notamment en datacenter, qui implémente l'*Enhanced Congestion Notification* pour la détection de problèmes de **congestion** dans le réseau pour en informer la couche transport : **TCP** peut alors adapter son débit en fonction.

3.1.5.4 Fragmentation

Lorsque qu'un datagramme est sur le point de passer via une couche 2 dont le **MTU** est trop petit, IP n'a que l'alternative suivante :

1. fragmenter le datagramme : si le bit DF (Don't Fragment) n'est pas activé, le routeur découpera le datagramme trop grand en petits morceaux dont chacun sera préfixé par un nouvel entête ; le *destinataire* seul défragmentera – s'il manque un fragment, le datagramme original entier doit être réexpédié.
2. ne pas fragmenter, détruire le datagramme et renvoyer une erreur **ICMP destination unreachable** de type *fragmentation needed* en laissant à l'émetteur la charge de résoudre le problème (voir ci-après).

Le champ d'identification (*Identifier*) contient la même valeur pour tous les fragments d'un datagramme original. Le bit DF (*Don't Fragment*) est prévu pour interdire toute fragmentation par les routeurs[18], tandis que le bit MF (*More Fragments*) indique que d'autres fragments

17. similaire au bourrage couche 2 mais pour une raison différente, voir section 2.4.1.3.2 en page 42
18. seul mode possible en IPv6

existent à des décalages supérieurs. Le champ de décalage indique la position du fragment dans le datagramme original ainsi fragmenté, de 0 à 8191 par unité élémentaire de 8 octets.

Pour des raisons de sécurité, car un déni de service (**DoS**) peut en résulter, les **firewall**s interdisent souvent le passage de fragments. Pour cette raison ainsi que pour simplifier le protocole, les émetteurs activent aujourd'hui en général le bit DF (Don't Fragment) et reposent sur le **PMTU DISCOVERY** pour détecter le **MTU** possible.

Le *Path MTU Discovery Mechanism* est la découverte, par essais successifs, du MTU maximum possible sur un chemin donné. Les routeurs intermédiaires informent l'émetteur des problèmes rencontrés grâce aux datagrammes **ICMP destination unreachable** de type *fragmentation needed*, et celui-ci informe les couches supérieures de l'adaptation.

3.1.5.5 Champ TTL

Le nom du champ **TTL** (*Time To Live*) induit en erreur : en pratique, il s'agit d'un (dé)compteur, initialisé à une valeur usuelle par l'émetteur du datagramme, qui est *diminué* d'une unité lors du passage à travers chaque routeur. Lorsque le champ passe à zéro, le datagramme est détruit et un message d'erreur ICMP (**ICMP time exceeded**, voir 3.1.6.2 en page 63) est généré par le routeur. L'effet est donc d'empêcher les boucles infinies de routage en limitant le nombre de sauts, ou de *hops* que peut faire un datagramme dans un réseau Internet [19]. Les implémentations IP mettent aujourd'hui 64 comme TTL par défaut initial (selon le RFC-1700) – MICROSOFT y met [20] 128.

La commande **traceroute** permet de découvrir la route parcourue par un datagramme jusqu'à sa destination grâce à des TTL croissants (voir page 64).

3.1.5.6 Options

Le(s) champ(s) optionnel(s) exist(ent) si la longueur de l'entête IP est plus grande que 5 (mots de 32 bits). Une option est toujours codée sur un multiple de 32 bits de la manière suivante :

code (8 bits)	length (8 bits)	data (jusqu'à 2 octets)
data (multiples de 4 octets éventuels ou **bourrage**)		

Le nombre d'octets de data est déterminé par le champ length. Le champ code se décompose en Flag Copy (1 bit), Class (2 bits) et Option Number (5 bits). Les fonctions importantes décrites dans le RFC-791 sont les suivantes :

copy	class	number	description	length
0	0	0	End of option list (RFC-791)	0
0	0	1	No operation (**bourrage** d'options)	0
1	0	2	Security (RFC-1108)	11
1	0	3	Loose source routing	variable
0	0	7	Record route	variable
1	0	9	Strict source routing	variable
0	2	4	Internet timestamp	variable

Certainess options, par exemple ci-dessus liées au **source routing** (pouvoir spécifier la route à suivre par un datagramme) ne sont en général plus supportées, pour des raisons de sécurité.

19. la valeur TTL définie par l'émetteur est le *diamètre maximum* d'Internet, exprimé en nombre de routeurs
20. ce qui permet déjà un problème de sécurité de type **fingerprinting** – être identifié par un attaquant

3.1.5.7 Protocole de couche supérieure

Le champ Protocole est utilisé pour identifier le protocole de couche supérieure (usuellement la couche transport) ou associé à IP qui envoie/reçoit les données utiles du datagramme. Les protocoles les plus courants sont décrits dans les tables suivantes :

Couche transport

code	protocole	description
6	TCP	Transmission Control Protocol (RFC-793)
17	UDP	User Datagram Protocol (RFC-768)
132	SCTP	Stream Control Transmission Protocol (RFC-4960, voir section 4.2.11.2 en page 90)
29	ISO-TP4	ISO Transport Class 4
46	RSVP	Resource Reservation Protocol (RFC-2205) de type **INTSERV**

Protocoles liés à IP

code	protocole	description
1	ICMP	Internet Control Message Protocol (RFC-792), voir section 3.1.6.2 en page 63
2	IGMP	Internet Group Management Protocol (multicasting), voir section 3.1.4.3 en page 57
103	PIM	Protocol Independent Multicast, voir section 3.1.4.3 en page 57

Protocoles de routage

code	protocole	description
3	GGP	Gateway-to-Gateway Protocol (RFC-823)
8	EGP	External Gateway Protocol (RFC-904), voir section 3.2.3.3 en page 70
9	IGP	Interior Gateway Protocol, voir section 3.2.3.2 en page 69
89	OSPF	Open Shortest Path Protocol (RFC-1010/1131), protocole de routage, voir 3.2.3.2.2 en page 69

Tunnels, VPNs et IPsec

code	protocole	description
41	**IPv6**	Tunnel IPv6 dans IPv4
47	**GRE**	Generic Routing Encapsulation (RFC-2784), utilisé pour certains **VPN** IP-sur-IP comme les échanges de paquets **PPTP**
50	ESP	**IPsec ESP** (*Encapsulation Security Payload*), paquet chiffré IPsec/ESP.
50	AH	**IPsec AH** (*Authentication Header*), paquet signé IPsec/AH.

3.1.6 Protocoles liés à IP

3.1.6.1 Protocole ARP

Comme les routeurs éliminent[21] les adresses couche 2 (MAC) avant la transmission pour ne conserver que les adresses réseau (IP) et éventuellement ajouter de nouvelles adresses couche 2 locales à la liaison concernée, il est nécessaire de recourir à des protocoles de correspondance d'adresses tels que **ARP**, **proxy-arp**, **RARP** et **BOOTP**.

Les protocoles de la famille ARP (*Address Resolution Protocol*) travaillent en couche 2 (couche liaison). La justification de cette multiplicité des adresses est principalement le fait que les adresses couche 2 (MAC) n'ont pas de structures hiérarchiques mais sont délivrées par constructeur. Un routage global sur la base de ces adresses serait un cauchemar : chaque adresse devrait être connue de tous les routeurs. Les adresses IP, elles, ont une structure hiérarchique correspondant plus ou moins à la géographie : des groupes de réseaux sont attribués aux continents ou aux pays. Les routeurs du reste du monde n'ont, dans un cas idéal, besoin que d'une entrée pour tout le groupe (en particulier en IPv6). Les adresses MAC ont donc une **portée** limitée à la liaison couche 2 qui les héberge.

Pour découvrir l'adresse MAC d'un équipement dont il ne connaît que l'adresse IP, une station ou un routeur doit utiliser le protocole ARP en envoyant un broadcast couche 2 (avec type Ethernet 0x0806) qui contient l'adresse IP connue et attendre en réponse l'adresse MAC de l'équipement qui aura reconnu son adresse IP. Un cache ARP avec un délai d'expiration relativement court permet d'éviter la congestion du réseau par les broadcasts ARP.

Le format d'une trame ARP (RFC-826) est le suivant :

<table>
<tr><td colspan="2">Entête couche 2
(p.ex : Ethernet Type=0x0806)</td></tr>
<tr><td colspan="2">Hardware (16)
(p.ex : 6 pour 802.x)</td></tr>
<tr><td colspan="2">Sender Protocol (16)
(selon type Ethernet)</td></tr>
<tr><td>Hard. Addr. Length (8)
(p.ex : 48 pour MAC Ethernet)</td><td>Prot. Addr. Length (8)
(p.ex : 32 pour IP)</td></tr>
<tr><td colspan="2">Opcode : Request=1/Reply=2 (16)</td></tr>
<tr><td colspan="2">Sender Hardware Address</td></tr>
<tr><td colspan="2">Sender Protocol Address</td></tr>
<tr><td colspan="2">Target Hardware Address (vide à la requête)</td></tr>
<tr><td colspan="2">Target Protocol Address (complété à la requête)</td></tr>
<tr><td colspan="2">Trailer couche 2D
(p.ex : Ethernet CRC/FCS)</td></tr>
</table>

Si elle a besoin de découvrir son adresse IP à partir de sa propre adresse MAC, une station peut utiliser, s'il est configuré, le protocole **RARP**[22] (*Reverse ARP*) qui travaille selon un principe très similaire à ARP. L'unique différence provient du fait que seule l'adresse MAC de l'émetteur est connue lors de la requête (broadcast avec Ethertype=8035h, Opcode=3). Ensuite, il peut y avoir plusieurs réponses (Opcode=4) de la part de serveurs RARP qui renvoient l'adresse réseau de l'émetteur en s'identifiant comme récepteur à l'aide de leur adresse réseau.

21. la portée (scope) des adresses MAC ne dépasse pas le domaine de diffusion (couche 2, ici correspond au sous-réseau) dans lesquelles elles sont utilisées.

22. le protocole DHCP (voir section 3.1.2.2 en page 52) est bien plus utilisé pour cela aujourd'hui

Pour compenser des erreurs de configuration de netmask ou pour la compatibilité avec de véritables ancêtres ne gérant pas les netmask, il est possible de configurer les routeurs en mode **proxy-arp**. Le principe est que le routeur va répondre à toute requête ARP pour une adresse couche 3 que le routeur sait router : il répondra en indiquant sa propre adresse MAC. Une utilisation plus moderne de ce protocole est par exemple pour émuler une couche 2 unique sur plusieurs liaisons interconnectées par un routeur : le but est de faire fonctionner des protocoles nécessitant la couche 2, comme par exemple le fameux voisinage réseau Microsoft à travers un réseau de couche 3[23]. Enfin, certaines applications comme les machines virtuelles avec réseau en mode bridge ont besoin du proxy-arp du côté de l'hôte ou d'un véritable switch logiciel.

3.1.6.2 Protocole ICMP

<table>
<tr><td colspan="3">IP Header (protocol=1)
(5x32 bits)</td></tr>
<tr><td>Type
(8)</td><td>Code
(8)</td><td>Checksum
(16)</td></tr>
<tr><td colspan="3">[paramètres]
(32)</td></tr>
<tr><td colspan="3">Information
(extrait éventuel des données utilisateurs du datagramme original)</td></tr>
</table>

Le protocole ICMP (*Internet Control Message Protocol*) a été prévu pour compléter le protocole IP avec des mécanismes de signalisation d'erreurs et de statuts de fonctionnement. Il s'agit d'un protocole non fiable en mode non connecté, bâti dans et au-dessus de la couche IP et donc difficilement plaçable dans le modèle OSI autrement qu'en couche 3.

Les types de message ICMP les plus courants sont décrits dans le tableau ci-après. Le checksum représente simplement le complément à 1 des champs Type et Code. Certains de ces messages ne sont plus aujourd'hui traités, pour des raisons de sécurité (notamment les messages de redirections) ou d'obsolescence.

type	**description**	**code**	**signification**	**raison**
8/0	Echo request/reply		ping/pong	commande `ping`
3	Destination unreachable	0 1 2 3 4	network unreachable host unreachable protocol unavailable port unreachable fragmentation needed	route manque ARP couche 4 MTU plus petit
5	Redirect datagrams	0-3	souvent bloqués	
11	Time exceeded	0 1	TTL arrivé à zéro pendant le transit fragment manquant pour réassemblage après délai	boucle de routage paquet perdu ou **DoS**

ICMP est à la base de deux outils très pratiques pour une première localisation de problèmes réseaux : **ping** et **traceroute**.

ping envoie de façon répétitive une requête ICMP 8 (**echo request**) puis mesure et affiche le temps jusqu'à l'arrivée de la réponse. L'adresse à atteindre peut être donnée directement (doted decimal notation) ou on peut fournir un nom qui sera résolu par DNS. La

23. par exemple un VPN

plupart des composants d'un réseau (serveur, PC, stations de travail, routeur, firewall...) répondent à ces requêtes ICMP. Néanmoins certains firewall bloquent ces messages car ils peuvent être utilisés pour des attaques. On peut grâce à cet outil contrôler si une adresse peut être atteinte. On mesure ainsi le délai aller-retour (**RTT**, *round trip time*).

traceroute (TRACERT.EXE sur les systèmes Microsoft) permet de suivre la route[24] utilisée pour atteindre une adresse. Cet outil fonctionne comme suit :

- une requête **ICMP echo request** (ou, avec certaines implémentations, un datagramme UDP à destination d'un port probablement libre au-dessus de 30'000) est envoyée à l'adresse souhaitée avec le champ **IP TTL** (time to live) à 1.
- ce datagramme est donc détruit par le premier routeur qui retourne un message **ICMP time exceeded** en y indiquant sa propre adresse IP (et un extrait du datagramme original)
- traceroute augmente ensuite le TTL (à 2, puis 3...) jusqu'à atteindre l'adresse désirée
- à chaque fois, l'outil affiche le temps de réponse et l'adresse du routeur qui a vu le TTL passer à zéro
- la station finale répond au message ICMP echo request, car celui-ci lui arrive avec un TTL= 1 (en alternative avec UDP : la station répond, si le port n'est pas utilisé par une application, par un message **ICMP port unreachable**, sinon, le datagramme est livré à l'application innocente ...).

3.1.6.3 DNS

Pour pouvoir travailler avec des adresses plus conviviales que ces adresses numériques, le SRI NIC (Network Information Center) a commencé par gérer un fichier appelé `HOSTS.TXT` qui contenait la liste des noms de réseau, routeur ou hôte correspondant aux adresses IP. Cette approche a rapidement dû être remplacée par une architecture *hiérarchique* de noms de domaines (**DNS**, *Domain Name System*) dont le premier niveau hiérarchique comprend 6+ domaines officiels (**TLD**, *Top-Level Domains*) :

TLD	rôle
arpa	ARPAnet (projet de recherche initial) : a encore un usage concernant la résolution inverse (champs **PTR**, domaine `in-addr.arpa` ou `ip6.arp`) et le plan de numérotation **E.164** lié à l'intégration de la téléphonie IP et classique (**ENUM**).
com	commercial
edu	éducation (USA)
gov	gouvernement et administration (USA)
int	organisations internationales
mil	militaire (US)
net	réseau
org	organisations à but non lucratif

A ces domaines de base ont été ajoutés :

- les codes **ISO-3166** des pays (p.ex. `ch` pour la Suisse, `fr` pour la France, `us` pour les Etats-Unis, etc)
- de nouveaux domaines (`info`, `biz`, `mobile`, ...) – il suffit de disposer d'un capital suffisant

24. à l'intérieur des réseaux d'opérateur **MPLS** `traceroute` peut ne pas montrer la réalité : l'option `--mpls` de la commande `mtr` permet, si le RFC-4950 est respecté, de voir également les routeurs intérieurs (**LSR**).

— parfois des domaines de portée limitée à quelques serveurs DNS (42).

Chacun de ces domaines peut encore posséder une hiérarchie de sous-domaines qui viennent en préfixe du domaine principal. Le nom `he-arc.ch` correspond par exemple au sous-domaine de la HE-Arc sous le TLD `ch`, tout comme `epfl.ch` ou `ragusa.ch`. A l'intérieur d'un sous-domaine peuvent se trouver des noms de machines situés ou non dans des sous-domaines (par exemple `gandalf.teleinf.labinfo.eiaj.ch`), représentant certain nombre de serveurs offrant différents services externes (HTTP, SMTP, FTP, ...) ou internes (POP3, SMB...).

Notons que la hiérarchie du DNS débute par un point mais il n'est pas obligatoire de mentionner ce point à la fin d'un nom de domaine : `www.he-arc.ch` est équivalent à `www.he-arc.ch.` – mais cela peut être utile de l'utiliser pour éviter que le domaine par défaut soit ajouté, notamment dans des fichiers de configuration de zone DNS.

3.1.6.3.1 Résolution DNS On remarque que le format RFC-822 pour la messagerie **SMTP** (voir section 7.2.5 en page 121), par exemple pour l'adresse `Marc.Schaefer@he-arc.ch`, est visiblement basé sur cette architecture. Pour envoyer un mail à cette adresse, un serveur de mail demandera simplement le nom (**MX**, *Mail eXchanger*), puis l'adresse IP (**A**) du serveur de mail distant à un serveur DNS **récursif** (sympathique), qui se chargera de trouver la réponse pour lui.

Ce serveur DNS récursif exécutera la procédure suivante :

1. demander à un des serveurs racines (dont la liste est configurée en dur) la résolution MX de `he-arc.ch`, qui lui indiquera qui gère le TLD `ch` (délégation **NS**)
2. demander au serveur gérant le TLD `ch` la même chose, il lui indiquera qui gère le domaine `he-arc.ch` (toujours une délégation)
3. demander à ce serveur la même chose, il lui donnera la réponse désirée, car il est **authoritative** (autoritaire) pour le domaine `he-arc.ch`.

Puis, si cette information n'a pas été communiquée avec la réponse MX, la même chose pour la requête A. Le serveur de mail pourra alors ouvrir une connexion TCP sur le service SMTP, puis envoyer l'e-mail.

3.1.6.3.2 Cache DNS En pratique, bon nombre de ces requêtes sont évitées par un mécanisme de **cache** présent sur tous les serveurs de noms, cache qui utilise un temps maximum de rétention, le temps de vie ou *Time to Live* (**TTL**[25]), spécifié par le serveur autoritaire gérant la zone concernée et décrémenté par les serveurs caches à chaque seconde : lorsque la valeur arrive à zéro, l'entrée est jetée et la prochaine requête demandant cette information sera recherchée à sa source autoritaire à nouveau.

3.1.6.3.3 Types de champs Voir la figure 3.2 en page 66.

Le DNS est aussi, aujourd'hui, impliqué dans la découverte automatique de services via les champs SRV, par exemple pour la voix-sur-IP (protocole **SIP**) :
`_sip._tls.domaine.ch IN SRV 100 1 443 sipdir.online.lync.com.`

ou dans la validation de propriété de domaine ou pour le stockage de clés publiques.

3.1.6.3.4 Outils Les outils de diagnostic du DNS sont `nslookup`, `host`, `dig` et `whois`. Quelques exemples d'utilisation suivent :

25. ne pas confondre avec le TTL d'IP, qui lui est un nombre de sauts et pas un temps.

A ou AAAA	nom vers adresses IPv4 ou IPv6
NS	délégation vers un autre serveur DNS
CNAME	alias : ne peut coexister avec d'autres RRs sur la même feuille ; de plus, il est interdit selon le RFC-1912 de faire pointer des MX, CNAME, PTR ou NS sur un CNAME
TXT	champ d'usage à définir : utilisé par exemple pour l'anti-spam **SPF**
PTR	résolution de nom inverse : adresse IP vers nom
MX	Mail eXchanger : serveur de mail ; ce champ, contrairement aux autres possède une priorité

FIGURE 3.2 – Types d'enregistrements (Resource Records) du DNS

```
schaefer@reliant:~$ host www.he-arc.ch
www.he-arc.ch is an alias for lo-ein-franklin.he-arc.ch.
lo-ein-franklin.he-arc.ch has address 157.26.64.89

schaefer@reliant:~$ host 157.26.64.89
Host 89.64.26.157.in-addr.arpa. not found: 3(NXDOMAIN)

schaefer@reliant:~$ host 157.26.77.13
13.77.26.157.in-addr.arpa domain name pointer gandalf.teleinf.labinfo.eiaj.ch.

schaefer@reliant:~$ host -t any he-arc.ch
he-arc.ch has address 157.26.64.89
he-arc.ch name server nestor.eicn.ch.
he-arc.ch name server smtpgw1.he-arc.ch.
he-arc.ch name server scsnms.switch.ch.
he-arc.ch has SOA record nestor.eicn.ch. netadmin.he-arc.ch. 642 28800 7200 604800 \
                                                              180
he-arc.ch mail is handled by 10 smtpgw1.he-arc.ch.
he-arc.ch descriptive text "v=spf1 mx ip4:157.26.64.0/24 -all"
```

On remarque qu'au moment d'exécuter la session ci-dessus, l'adresse IP 157.26.64.89 n'avait pas de **reverse** (pas de nom, champ **PTR**), ce qui peut parfois poser des problèmes (en particulier pour le SMTP, SSH et le FTP), et qu'un champ spécial de type **TXT** existe permettant d'améliorer l'anti-spam en interdistant à quiconque d'émettre des mails avec un expéditeur sous he-arc.ch s'il ne provient pas du sous-réseau indiqué – du moins pour les serveurs supportant le protocole **SPF**.

3.1.6.3.5 Haute fiabilité et répartition de charge avec le DNS

Le service de mail de Google dispose de deux enregistrements MX (avec le plus prioritaire la 2e ligne avec la valeur 20 – **HA**, *High Availability*). Quant à son service web, les deux enregistrements A sont en répartition de charge (**load balancing**) – tourniquet, **round-robin**.

```
schaefer@reliant:~$ host -t mx google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.

schaefer@reliant:~$ host -t a www.google.com.
```

```
www.google.com has address 173.194.35.16
www.google.com has address 173.194.35.19
```

3.1.6.3.6 Application du TTL du DNS : DynDNS Une application du **TTL** (voir section 3.1.6.3.2 en page 65) est la possibilité de supporter des adresses IP dynamiques :

```
schaefer@reliant:~$ dig shakotay.dyn.alphanet.ch. @8.8.8.8 | grep '^sh'
shakotay.dyn.alphanet.ch. 12    IN      A       89.217.235.249
```

Le TTL sur l'enregistrement `shakotay.dyn.alphanet.ch` restant est très court (le cache sera invalidé dans 12 secondes). Si l'on interroge un des serveurs **autoritaire**s gérant la zone concernée directement, on obtiendrait la valeur initiale qui est de 30 secondes. Il existe des services publics comme **DynDNS**.org.

3.1.6.3.7 Autorité Cela nous permet de définir la notion de serveur **autoritaire** : ci-dessous la requête est envoyée au serveur DNS configuré par DHCP à la HE-Arc (qui est un simple **cache récursif**, donc répondant aux requêtes de clients complètement) :

```
schaefer@reliant:~$ nslookup shakotay.dyndns.org
Server:         157.26.80.30
Address:        157.26.80.30#53

Non-authoritative answer:
Name:   shakotay.dyndns.org
Address: 89.217.235.249
```

Et ci-dessous la requête est envoyée directement au serveur autoritaire concerné :

```
schaefer@reliant:~$ nslookup shakotay.dyndns.org ns1.dyndns.org
Server:         ns1.dyndns.org
Address:        204.13.248.75#53

Name:   shakotay.dyndns.org
Address: 89.217.235.249
```

La notion d'autorité est définie dans le champ `SOA`, et, en pratique les délégations de sous-domaines (via le champ `NS`).

3.1.6.3.8 Risques du DNS Le DNS est par défaut un protocole en clair, sans authentification : sans **DNSSEC** ou un transport partiellement sécurisé (**DoT** ou HTTPS – **DoH**), seuls le numéro de port du client, le numéro de requête et éventuellement la casse permettent d'assurer que la réponse vient bien du serveur interrogé.

De plus, il peut être soumis à des attaques de pollution de cache (en particulier si un serveur DNS est à la fois autoritaire et récursif), ou, comme le NTP (*Network Time Protocol*), il peut être une source d'attaques **DDoS**, voir section 3.4.3 en page 81.

3.2 Routage

3.2.1 Principes

La fonction principale d'une couche 3, en particulier dans le modèle léger d'IP, est le routage des datagrammes à travers le réseau.

Comme déjà mentionné, les routeurs différent des ponts (bridges, commutateurs, switches) dans le sens où ils travaillent en couche réseau (3). Leur rôle consiste, pour chaque protocole réseau, à choisir une route en fonction d'informations contenues dans des tables de routage définies par les administrateurs de réseau (manuellement, voir section 3.2.2 en page 68 ou automatiquement et selon des règles d'optimisation par des protocoles dits *de routage* voir section 3.2.3 en page 69).

Le routage est toujours effectué du plus précis au plus global, pour éviter les ambiguités.

3.2.2 Routage statique

3.2.2.1 Introduction

Dans les cas les plus simples et en particulier pour les routeurs et serveurs, on utilisera du routage statique préconfiguré à l'installation de la machine (ou par un serveur DHCP, voir section 3.1.2.2 en page 52).

3.2.2.2 Exemples

Voici par exemple l'état de la table de routage d'un client situé derrière un routeur/NAT avec serveur DHCP :

```
schaefer@reliant:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 eth0
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
```

On y observe trois routes, dont la plus précise est 192.168.1.0/24 via l'interface eth0, et la moins précise (la **route par défaut**) est celle qui sera utilisée lorsqu'aucune autre ne s'applique (0.0.0.0/0 via 192.168.1.1). On y remarque aussi une route d'autoconfiguration vers **169.254.0.0/16** (voir section 3.1.3.7 en page 56), et l'absence d'une route vers le **loopback** car cette règle est automatique et transparente sur cette plateforme.

Un autre exemple plus complexe d'un serveur/routeur/NAT/firewall avec beaucoup d'interfaces, plusieurs adresses IP et une configuration en multi-homing statique (voir section 3.2.4 en page 71) :

```
schaefer@reliant:~$ ssh root@virtual ip -4 route show
195.141.51.211 via 192.168.100.2 dev eth2
192.168.99.0/24 dev venet0 scope link
212.35.56.33 via 192.168.100.2 dev eth2
83.68.223.48/28 dev eth1  proto kernel  scope link  src 83.68.223.60
```

```
192.168.100.0/24 dev eth2  proto kernel  scope link  src 192.168.100.1
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.11
192.168.201.0/24 dev eth3.0000  proto kernel  scope link  src 192.168.201.1
192.168.200.0/24 dev eth3.3  proto kernel  scope link  src 192.168.200.1
192.168.250.0/24 dev tap0  proto kernel  scope link  src 192.168.250.254
192.168.202.0/24 dev eth3.4  proto kernel  scope link  src 192.168.202.1
80.83.54.0/24 dev eth1  proto kernel  scope link  src 80.83.54.2
default via 83.68.223.49 dev eth1  src 83.68.223.60
default via 80.83.54.1 dev eth1

schaefer@reliant:~$ ssh root@virtual ip -4 route show table 4
default via 192.168.100.2 dev eth2
```

3.2.3 Protocoles dynamiques de routage

3.2.3.1 Motivation et fonctionnement général

Lorsque le réseau est vaste et en topologie maillée, la principale difficulté est sans doute la détermination du critère permettant de choisir la meilleure route pour atteindre la destination. L'idée la plus simple consiste à choisir la route qui nécessite le moins d'étapes (**hop**s), tandis que les méthodes les plus évoluées incluent des critères tels que le coût du service, le débit offert, la sécurité proposée et l'état de la liaison (appelés facteurs de qualité de service – **QoS** dans le modèle OSI). Ces facteurs évoluent bien entendu au cours du temps, si bien que les routeurs doivent recourir à des méthodes adaptatives et dynamiques pour mettre à jour leurs tables de routage.

Pour ce faire, les routeurs s'échangent périodiquement leurs informations à l'aide de messages dédiés de protocoles de **routage**.

3.2.3.2 Internes (IGP)

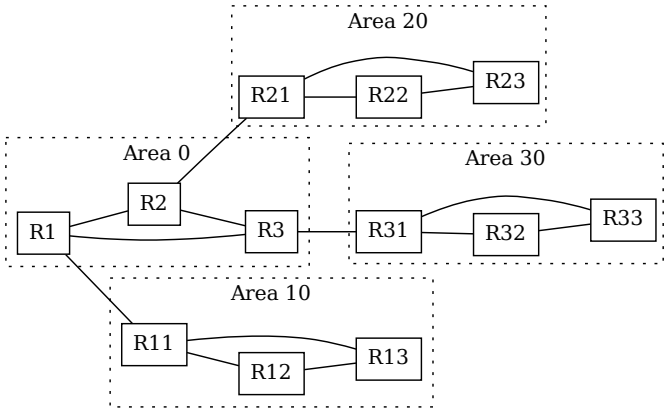
Les protocoles de routages dits internes (*Interior Gateway Protocol*) sont exploités sur les routeurs dits *intérieurs* utilisant des protocoles d'échange interzones mais d'une portée limitée à une organisation (RIP-gated, OSPF, BGP...) éventuellement propriétaires (Cisco **EIGRP**).

3.2.3.2.1 RIP Le *Routing Information Protocol* version 2 (RFC-2453, STD-56) est un protocole de routage basé sur un algorithme de vecteur distance, mesuré par le nombre de sauts (**hop**s). Les routeurs échangent régulièrement les informations de routage puis décident localement. Ce protocole, même dans sa version 2 publiée en 1998, souffre de diverses déficiences (diamètre maximum de 15 sauts, topologie plate, temps de synchronisation...) dont certaines sont corrigeables par des options.

3.2.3.2.2 OSPF *Open Shortest Path First*, version 2, publié dans le RFC-2328 est un protocole très utilisé pour le routage interne à une organisation (à l'intérieur d'un seul système autonome, **AS**, voir 3.2.3.3.2 en page 70). Il construit un arbre d'accessibilité global en fonction des informations échangées, puis trouve l'arbre des chemins minimaux suivant l'algorithme de **Dijkstra**.

Ses avantages principaux sur RIPv2 sont :

- temps de convergence très court, en particulier en cas de panne d'une liaison (inondation)
- vecteur distance basé sur plus de paramètres (coût : délai **RTT**, débit, disponibilité, fiabilité, etc)
- structurable en zones (*areas*), dont la zone zéro est le **backbone**, ce qui permet de limiter les échanges de tables de routage
- utilise le **multicast** par défaut (224.0.0.5)
- peut fonctionner de manière sécurisée (authentification)



3.2.3.3 Externes (sur Internet, EGP)

Les protocoles de routage dits externes sont quant à eux exploités sur des routeurs dits *extérieurs* ou *de bords* (*Border Gateway* ou *Boundary Routers*) utilisant des protocoles d'échange interréseau (GGP/core, EGP...) nécessairement supportés par les différents équipementiers.

Ces équipements doivent bénéficier de performances importantes (mémoire, CPU et réseau) : le volume des tables de routage, le temps de recherche dans celles-ci et le trafic engendré par ces échanges ne peut pas être négligé (en particulier car certains protocoles échangent toute l'information plutôt que seuls les changements survenus).

L'ensemble des réseaux interconnectés par ces routeurs forme alors un réseau global dont le plus célèbre exemple est certainement Internet.

3.2.3.3.1 BGP Le protocole BGP4/CIDR (*Border Gateway Protocol 4, Classless Inter-Domain Routing*) essaye de réduire la taille des tables de routage en se basant sur l'agrégation de route (voir section 3.1.3.4 en page 54) pour ne créer qu'une seule information de routage pour la gestion de plusieurs réseaux d'adresses adjacentes allouées à un même site.

3.2.3.3.2 Autonomous Systems (AS) Le concept de système autonome (AS) est essentiel dans l'Internet global pour la gestion des tables de routage. Il s'agit, selon Wikipédia, d'un ensemble de réseaux informatiques IP intégrés à Internet et dont la politique de routage interne (routes à choisir en priorité, filtrage des annonces) est *cohérente*.

Cela signifie qu'au sein d'un AS, on utilisera en général un protocole de routage interne (voir section 3.2.3.2 en page 69), donc que l'entité administrative gérant un AS doit être généralement unique (p.ex. une entreprise en multi-homing ou un fournisseur d'accès à Internet). Entre AS, on utilisera des protocoles externes.

Un AS est identifié par un petit entier sur 16 bits, l'**ASN** (*Autonomous System Number*). Ce numéro est affecté par les mêmes organisations qui allouent les adresses IP (**RIR**s). Les numéros entre 64512 et 65534 sont réservés pour un usage personnel, et ne doivent pas être utilisés pour un réseau relié à Internet. Une révision importante de la numérotation des AS, passant de 16 à 32 bits (de manière relativement rétro-compatible pendant la période de transition, voir le RFC-4893) a été mise en place en 2007 vu l'augmentation exponentielle de ces derniers.

3.2.3.4 Routage de flux et MPLS

D'autres algorithmes de routage peuvent être mis en oeuvre, en particulier lorsque les protocoles de routage coûtent trop cher, ne sont pas assez flexibles, ou prennent trop de temps à changer d'avis (converger).

Citons par exemple le routage local (CISCO VRF) effectué par les routeurs non plus sur l'adresse destination mais sur des critères[26] comme le type de flux ou une marque valable uniquement au sein d'un réseau particulier. En fait, ce routage de flux est un cas particulier du concept de routage par **label** (étiquette) utilisé déjà dans **ATM** et aujourd'hui dans **MPLS** : les routeurs de bord de MPLS (**LER**, *Label Edge Router*s) utilisent le routage de flux pour classifier le trafic au bord du réseau MPLS et affectent une étiquette (**label**) à chaque paquet, étiquette qui permet ensuite la commutation rapide au sein du réseau MPLS par les **LSR** (*Label Switching Router*s) – voir un exemple d'architecture de réseau core MPLS à la figure 3.3 en page 82.

Ces réseaux **MPLS** (*Multi Protocol Label Switching*) sont très souvent exploitées pour le cœur de réseau haute performance des opérateurs (réseau **core**) en intègrant la **qualité de service**. Leur commutation très rapide en couche 2 du protocole IP (v4 et/ou v6) de couche 3 est la raison de leur succès.

3.2.3.5 Routage de la patate chaude

Lorsqu'une ligne est surchargée, la meilleure décision de routage du point de vue de la topologie, du débit ou d'autres critères est inefficace : on risque de maintenir les datagrammes longtemps dans des queues avant de pouvoir les transmettre. Une tactique consiste à envoyer ce datagramme sur n'importe quelle interface libre, même si cela l'éloigne éventuellement de la destination. On espère que le routeur au bout de cette ligne pourra peut-être prendre une meilleure décision.

Une autre raison est, que plutôt effectuer des décisions de routage complexes dans son réseau, de transmettre ces décisions à quelqu'un d'autre que l'on suppose mieux connecté. Cela peut mener à des décisions de routage moins efficaces dans certains cas, à un coût résiduel plus bas.

3.2.4 Multi-homing

On entend par multi-homing le fait d'héberger un service accessible par plusieurs adresses IP, fournisseurs, AS et/ou liaisons différentes, en général pour des raisons de sécurité (haute fiabilité, *High Availability*, **HA**) ou de performance (répartition de charge ou **load balancing**).

La méthode la plus simple est de connecter le service à deux liaisons différentes sur deux fournisseurs différents, et d'obtenir au moins une adresse IP fixe pour chacune des liaisons. On configurera ensuite un ou plusieurs équipements réseau (firewall, load-balancer, serveur proprement dit) avec ces deux adresses. Cette méthode ne nécessite pas de plage d'adresse spécifique, mais nécessite des configurations DNS spécifiques (la plus simple étant le **round-robin** – voir section 3.1.6.3.5 en page 66 – et la plus complexe étant des caches DNS répartis autour du globe et adressés par des adresses quasi **anycast** et retournant des enregistrements DNS différents suivant l'émetteur de la question : c'est une des méthodes que de nombreux **CDN** utilisent en plus du BGP ci-dessous).

Une méthode plus avancée consiste à gérer soit même le routage **BGP** et l'annonce d'un Autonomous System (AS) spécifique sur les deux liaisons, pour une plage d'adresse indépendante

26. **SRv6**, *Segment routing*, voir https://cisco.app.box.com/v/Segment-routing-JRES2017

du fournisseur (**plage PI**). Les routes seront alors choisies en fonction de règles statiques (filtres, politiques de passage de trafic) mais aussi de l'état des liaisons.

Par exemple, pour la HE-Arc (et d'autres institutions neuchâteloises) :

```
schaefer@reliant:~$ whois -h whois.ripe.net 157.26.0.0/16
inetnum:         157.26.0.0 - 157.26.255.255
netname:         ETNA
descr:           Haute Ecole Arc
descr:           formerly EICN
descr:           Le Locle, Switzerland
country:         CH
admin-c:         AM255
tech-c:          CW115
tech-c:          MM2424-RIPE
tech-c:          AJ2404-RIPE
status:          ASSIGNED PI
mnt-by:          SWITCH-MNT
[ ... ]
route:        157.26.0.0/16
descr:        ETNA
origin:       AS559
mnt-by:       AS559-MNT
source:       RIPE # Filtered
```

Noter l'état `ASSIGNED PI` (*provider independant*) et l'ASN 559. Attention : ne pas déduire de l'exemple ci-dessus que tous les AS gèrent des anciennes classes A-C.

On peut ensuite interroger une base de données BGP d'un serveur aux Etats-Unis pour voir quelles sont les routes possibles vers cet AS :

```
schaefer@reliant:~$ telnet route-server.he.net
Trying 64.62.142.154...
Connected to route-server.he.net.
Escape character is '^]'.
route-server> show ip bgp 157.26.0.0/16
BGP routing table entry for 157.26.0.0/16
Paths: (33 available, best #17, table Default-IP-Routing-Table)
  Not advertised to any peer
  559
    91.206.52.53 from 216.218.252.159 (216.218.252.153)
      Origin IGP, metric 20, localpref 100, valid, internal
      Originator: 216.218.252.153, Cluster list: 216.218.252.159 216.218.252.174
      Last update: Tue Nov  8 00:20:31 2011
[ ... ]
```

3.2.5 Routage et anti-spoofing

IP ne vérifie pas les adresses sources (il n'y a pas d'authentification des entêtes, à part si **IPsec AH** est utilisé) : le fait d'émettre des datagrammes dont l'adresse source est fausse s'appelle le **spoofing**. La technique la plus simple pour se prémunir contre le spoofing est basée sur les

tables de routage. On autorisera un datagramme portant une adresse source A provenant d'une interface donnée que s'il existe une règle de routage concernant l'adresse A et qui routerait vers l'interface concernée.

Une telle technique fonctionne le mieux *au plus près* de la source : sur l'équipement d'agrégation de trafic provenant de clients. Elle ne pose problème que dans des cas particuliers (**routage triangulaire** des **VPN**s ou de **Mobile IP**, certains cas de multi-homing, adressage privé, etc). Elle n'est malheureusement pas implémentée par tous les fournisseurs[27] : se baser sur des adresses sources dans l'Internet global pour l'authentification est donc dangereux, en particulier pour les protocoles basés sur une couche 4 UDP (sans connexion), en plus du risque de protocoles potentiellement sources de **DDoS**[28] comme le DNS ou le NTP.

3.2.6 Translation d'adresse (NAT)

3.2.6.1 Principes

Un routeur normal ne change jamais les adresses couche 3 dans les datagrammes transmis : la translation d'adresse consiste en du routage avec modifications d'adresse(s) couche 3 (NAT) voire couche 4 (**PAT**). Cette translation est basée sur l'interface réseau d'entrée ou de sortie, le contenu des paquets (type de trafic, adresses, numéros de ports, protocoles de couches supérieures, qualité de service...) ainsi que sur l'éventuel état interne du routeur avec NAT (**stateful connection tracking**). Elle s'applique autant aux flux connectés (TCP) que non connectés (UDP), ainsi que p.ex. à ICMP.

Le cas général, fort rare, consiste en une translation d'adresse de N machines internes vers M adresses externes. Le NAT est aujourd'hui surtout utile pour cacher derrière une ou plusieurs adresses IP publiques M, souvent 1 (NAT N–1, *many-to-one*), un certain nombre de machines N en adresses IP privées, soit pour simplifier la configuration ou pour des raisons de disponibilité d'adresses. Un dernier cas dégénéré est le NAT 1–1 (*one-to-one*, ne nécessitant pas d'état dans l'équipement : **stateless**), souvent appelé *mode DMZ* (de manière erronée car sans offrir l'iolsation d'un vrai **DMZ**) dans les petits routeurs ADSL.

L'évolution des standards IPv4 en fonction de la raréfaction d'adresse est passée par les étapes suivantes :

- 1993 : RFC-1519 (adressage CIDR, permettant d'éviter le gaspillage induit par les anciennes classes)
- 1994 : RFC-1631 (NAT)
- 1996 : RFC-1918 (adresses privées)
- 2000 : RFC-3021 (/31 géré comme un cas particulier pour les liaisons point à point)
- 2012 : RFC-6598 (Carrier Grade NAT et plage 100.64.0.0/10)

La suite de l'évolution sera probablement l'usage d'IPv6 au sein du réseau de transport de l'opérateur, avec des adresses privées IPv4 fournies à l'utilisateur final, voire même d'adresses privées IPv4 pour les équipements du **backhaul**[29] comme on le voit de plus en plus (source : mailing-list SWINOG).

27. et pourtant, c'est une bonne pratique, voir le **BCP-38** http://bcp38.info.
28. Distributed **DoS** : déni de service distribué sur Internet.
29. réseau d'amenée entre l'utilisateur final et le réseau **core** du fournisseur

3.2.6.2 Passage d'un NAT

La plus grande difficulté du NAT est le problème du **NAT traversal**, en particulier avec l'usage de protocoles de couche application qui font usage d'adresses de couche 4 ou 7 : par exemple les services d'annuaires de services. Mais la modification du nexus implicite en cas d'usage d'adresses privées et de NAT/PAT peut également poser problème.

3.2.6.2.1 Collision de nexus L'usage le plus commun aujourd'hui étant le NAT N–1, il est donc nécessaire de stocker un état interne, par exemple basé sur des flux vus précédemment, sous forme d'une table de connection tracking avec une durée de vie, pour pouvoir identifier les transformations à effectuer à l'aller et au retour.

En NAT N–1, si deux machines internes en adresses privées créent chacune une connexion TCP ou un flux UDP vers le même serveur, le même numéro de port serveur et par malchance depuis le même numéro de port client, provoquant une collision de **nexus** (voir section 4.1.2 en page 84), le PAT allouera alors automatiquement un nouveau port inutilisé dans une plage supérieure et effectuera les modifications d'entêtes nécessaires. C'est un peu plus compliqué pour des protocoles comme ICMP [30] : dans ce cas on doit soit utiliser l'association à un flux de couche 4 existant auquel se réfère le datagramme ICMP, soit utiliser certains champs spécifiques de son entête pour l'identification du flux et donc de la machine interne concernée.

3.2.6.2.2 Annuaires de services La modification décrite ci-dessus, nécessaire en cas de collision de nexus, crée elle-même un problème pour tous les protocoles de couche 7 qui utilisent des numéros de port. Par exemple, un client de voix-sur-IP pourrait vouloir s'enregistrer auprès d'un proxy SIP pour être *appelé* ensuite par le même flux UDP [31].

Additionnellement, si la correspondance n'est pas maintenue en permanence (souvent elle s'efface, sans trafic, après 2-3 minutes), alors cela ne fonctionnera plus. On utilise pour ce faire des options de **keep-alive** des protocoles de couche 7.

Mais s'y ajoute le problème que le client ne connaît pas son bon numéro de port. Le serveur le connaît, mais beaucoup de protocoles de couche 7 précisent en couche 7 l'association. Le routeur NAT/PAT doit en plus, pour certains protocoles (SIP et FTP par exemple) corriger les adresses de couche 4 spécifiés dans l'entête couche 7 ! Cela ne fonctionne bien évidemment plus dès le moment que du chiffrement ou de l'authentification des entêtes sont utilisés.

Un serveur extérieur **STUN** peut alors être utilisé (avec un protocole UDP) pour obtenir le **nexus** réellement utilisé, si le client et son protocole de couche 7 supporte ce protocole.

En pratique, lorsqu'un téléphone SIP (voix-sur-IP) démarre, il va contacter un service d'annuaire pour s'enregistrer et associer son numéro de téléphone à son adresse IP et son numéro de port. Si le service d'annuaire et le téléphone sont séparés par un NAT, le NAT doit également corriger l'adresse IP, voire le numéro de port dans la demande d'enregistrement couche 7 et autoriser les flux entrants vers le téléphone (ce qui a également un impact sur la sécurité, s'il s'agit d'un firewall/NAT). L'usage de chiffrement rend impossible cette détection et manipulation par le routeur NAT et nécessite la collaboration des équipements, par exemple à l'aide du protocole **STUN**, implémentée grâce à des serveurs publics.

Consulter également la section 3.2.7 en page 75 pour plus de détails sur le passage de firewalls et de NATs.

30. voir RFC-5508 et RFC-3022
31. c'est plus facile en UDP qu'en TCP, car les firewalls bloquent en général les connexions entrantes TCP, voir section 3.2.7 en page 75

3.2.6.2.3 Carrier Grade NAT (CGNAT) Les fournisseurs d'accès font un usage assez important du NAT (**CGNAT**, *Carrier Grade NAT*), en particulier pour les mobiles, vu la difficulté d'obtenir des plages d'adresses publiques IPv4 libres, le déploiement encore préliminaire de l'IPv6 et le nombre de terminaux à connecter.

3.2.7 Pare-feu (firewall)

3.2.7.1 Définition

On peut voir un **firewall** comme un élément d'isolation de réseau, un routeur qui décide, sur la base de règles de sécurité, de ne pas router. Evidemment, les firewalls modernes prennent leur décision en examinant l'ensemble des couches (en particulier 2, 3, 4, voire 7) et peuvent valider les datagrammes au sein de flux préalablement établis.

3.2.7.2 Passer les firewalls

Un firewall, combiné ou non avec un NAT/PAT, va compliquer les échanges, en particulier dans le cas d'applications **P2P** (*peer-to-peer*). On utilisera alors une ou plusieurs techniques :

- utiliser UDP et non pas TCP, puis utiliser un serveur STUN : fonctionne si le NAT n'est pas symétrique [32], puis publier les adresses et ports publics sur un serveur central ou un annuaire
- ou utiliser un serveur central agissant comme relais (de connexions TCP ou de flux UDP) : l'idée est que tous les clients envoient et reçoivent du serveur central qui *commute* les messages entre clients au besoin, ce qui crée du délai
- faire des trous manuels dans le firewall ou via **UPnP** [33], si supporté (c'est évidemment un risque de sécurité)
- enfin un VPN peut lier les différents clients (via un serveur central, éventuellement dans le cloud)

Le **VPN** peut être la solution simple pour s'affranchir de problèmes d'adressage et de firewall / NAT. La seule difficulté importante en IPv4 est qu'il faut éviter d'avoir les mêmes sous-réseaux privés dans le VPN que pour le réseau local routé en NAT.

3.2.7.3 Communication directe entre clients web (navigateurs)

Le **WebRTC** (API HTML5 Javascript) peut être utilisé pour échanger en TCP ou UDP (plus facile, voir section précédente) directement entre navigateurs sans serveur relais. C'est très avantageux du point de vue du délai, mais ce n'est pas toujours possible.

La communication directe n'est toutefois pas toujours possible (firewall, mode **client isolation** du point d'accès WiFi) : dans ce cas, on pourra utiliser des **Websocket**s ou un VPN.

Les **Websocket**s permettent d'implémenter un relais efficace via le serveur web. On peut aussi, pour des protocoles **M2M** (*machine-to-machine*) se borner à relayer des messages : passer par un relais HTTP codé pour l'occasion ou, en plus complexe et flexible, un serveur MQTT [34].

32. https://wapiti.telecom-lille.fr/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2005ttnfa2006/butin-sutter/description_nat_firewall/description_nat_firewall.htm#nat_symetrique et https://fr.wikipedia.org/wiki/Simple_Traversal_of_UDP_through_NATs

33. permet d'agir sur les firewalls locaux, voir https://en.wikipedia.org/wiki/Internet_Gateway_Device_Protocol, par exemple sous Linux avec les outils http://miniupnp.free.fr/ : `upnpc -a 192.168.1.12 139 4139` va NATer le port 4139 du routeur vers le port SMB/139/TCP du client 192.168.1.12

34. https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment

3.3 Autres implémentations et avenir des réseaux

3.3.1 Autres implémentations

De nombreux protocoles ont été développés pour la couche 3, notamment au sein du modèle OSI : citons par exemple X.25 (réseaux de téléinformatique), dans le domaine de la téléphonie les signalisations **Q.931** (en bordure du réseau) et **SS7** (inter-opérateurs) ou encore Novell **IPX** dans les LAN.

Force est de constater que malgré le transport d'IP dans des réseaux mixtes couche 2/couche 3 notamment pour les **réseaux core** des opérateurs (**MPLS** – voir section 3.2.3.4 en page 71), IP reste le protocole universel le plus déployé.

3.3.2 IP version 6 (IPv6)

3.3.2.1 Historique

IPv4 montrant ses limites, notamment d'adressage, mais aussi pour la performance de routage lié à la fragmentation de l'espace d'adressage [35], au début des années 90, une mise au concours pour définir la version 6 [36] du protocole IP a été lancée par l'IETF. Une des propositions a été retenue en 1994 et elle a été affinée par la communauté, via les RFCs, pour être quasi finalisée en 1998 dans le RFC-2460. Des RFCs sont toutefois constamment publiés et mis à jour ou complètent des aspects spécifiques (p.ex. API socket et DHCPv6 en 2003, mobilité et flux en 2004, adressage en 2006).

Une journée de test a été organisée en 2011 et, en 2012, une activation globale a eu lieu : depuis IPv6 est entré dans sa phase productive [37], au côté d'IPv4.

Aujourd'hui, la version actuelle du protocole IP reste la 4, mais la version la plus récente est la 6, qui tourne en parallèle à la v4 et est disponible presque partout aujourd'hui (parfois non nativement). Même si de nombreux problèmes d'IPv4 ont été résolus, du moins en partie, IPv6 apporte beaucoup d'améliorations, mais en même temps la migration reste lente, avec peu de services spécifiquement disponibles en IPv6 [38].

3.3.2.2 Adressage IPv6 et transition d'IPv4

Les nouvelles adresses IPv6 sont codées sur 16 octets (128 bits), ce qui conduit à un nouvel entête IP simplifié sur 40 octets au lieu des 20 octets d'IPv4.

Pour la transition de clients [39] vers v6, plusieurs possibilités existent :

- en v4 seulement, utilisation d'un proxy pour l'accès v6
- utilisation de tunnels v6 sur v4 lorsque les routeurs proches n'ont pas l'IPv6 (moins performant, délai)
- en natif et en **dual-stack** : IPv4 et IPv6 coexistent partout

35. mais aussi car l'entête n'est pas de taille fixe en v4, ou en raison d'entêtes de signification peu précise, des checksums inutiles, le mauvais support de certaines fonctionnalités comme l'authentification, le chiffrement et le multicast, la complexité de la configuration et l'absence d'anycast
36. une expérimentation précédente, RFC-1819, ayant déjà utilisé la version 5
37. https://www.ipv6.com/general/ipv6-the-history-and-timeline/
38. à l'exception du protocole **onioncat** de tor, ou **DSLite** qui fournit des adresses v4 privées en **CGNAT**, mais des v6 publiques sur lesquelles ont peut déployer des serveurs
39. pour des serveurs, les mêmes possibilités existent : le proxy sera simplement un **reverse proxy**

— utilisation de v6 seulement, avec du **NAT64** (et un support dans le DNS) ou un proxy pour l'accès v4

Notamment, un plan de transition appelé **SIT** (*Simple IPng Transition*) prévoit de convertir les stations en dual-stack, compatibles avec les versions v4 et v6. Les adresses v6 initiales seront simplement les adresses v4 précédées d'un préfixe (12 octets), ce qui devrait permettre de convertir (encapsuler) rapidement les adresses v6 au format v4 et vice versa.

Les opérateurs peuvent déployer l'IPv6 de différentes manières, que cela soit en natif (dual-stack y compris sur les routeurs), SIT, des tunnels IPv6 sur IPv4 (avec délai dû au relais de datagrammes !), ou **IPv6rd** (*Rapid Deployment*, tunnel dont le relais est proche du premier routeur du fournisseur et la configuration simplifiée).

3.3.2.3 Changements principaux

Les modifications principales sont les suivantes :

- adresses (beaucoup !) plus grandes (128 bits)
- nouveau schéma d'adressage (plusieurs hiérarchies possibles, notion de **portée** – **scope** en anglais)
- généralisation du concept CIDR (avec les sous-réseaux les plus petits des /64, en principe)
- simplification de l'entête IP (taille fixe) et concept de chaînage pour des cas particuliers (chiffrement ou fragments p.ex.)
- suppression de la fragmentation par les routeurs et définition d'une taille minimum de datagramme garantie supportée par le réseau
- adressage amélioré
  - une machine dispose toujours d'au moins une adresse automatiquement configurée : au minimum une adresse de portée couche 2 (portée lien, **link-local**), permettant d'atteindre uniquement les machines branchées à cette interface, est auto-configurée dans tous les cas avec le bas de l'adresse (la partie hôte) choisi (voir ci-après)
  - pour atteindre Internet v6, le minimum nécessaire est une adresse IPv6 de **portée globale** et l'adresse d'un routeur accessible via la couche 2 ; les options de configuration sont
    - entièrement manuellement, comme en v4
    - si un serveur d'annonce de préfixe et du routeur par défaut (**RA**, *Router Advertisement*) existe, alors la machine construira son adresse composée du préfixe annoncé (64 bits) et le bas de l'adresse (la partie hôte) sera choisi comme ci-après
    - si un serveur DHCPv6 existe, l'adresse sera configurée par ce serveur similairement à v4
  - la *partie hôte de l'adresse* (les 64 bits du bas) est donc soit attribuée manuellement, par DHCPv6, ou formée de l'adresse MAC étendue à 64 bits (**EUI-64**, ou, pour des raisons de protection de la sphère privée et éviter l'**IP-tracking**, une adresse aléatoire)
  - hiérarchisation diminuant potentiellement la taille des tables de routage
- remplacement d'ARP par un nouveau protocole basé sur du multicast ICMP : **NDP** (*Network Discovery Protocol*), qui sert aussi à tester l'unicité des adresses
- suppression des broadcasts (remplacés par le multicast, dont le support est obligatoire)
- meilleur support pour les données multimédia (à la fois dans le multicast et la qualité de service (**QoS** et de label de flux)
- besoin moindre de NAT et possibilité d'utiliser des adresses de portée **site-local** (privées) spécifiques à chaque site et donc sans risque de collisions d'adresses en cas de fusions

de sites ou de réseaux privés
— support du chiffrement IPsec natif, de la mobilité IP...

IPv6 amène donc des simplifications pour le routage, pour la gestion administrative et la migration de services, notamment.

3.3.2.4 Compatibilité des applications

Même si les couches supérieures ne subissent pas de changements importants [40], toute application manipulant des adresses devra, à terme, être modifiée. Ce processus est en cours et des confinements de compatibilité sont évidemment possibles durant la transition.

3.3.3 Routage par oignon

Même s'il ne s'agit pas strictement d'une couche 3 [41], **tor** (*The Onion Router*) permet de créer des connexions à travers Internet en assurant un certain anonymat.

En effet, tor brouille la **traçabilité** en ajoutant 3 intermédiaires (relais) entre un client et un serveur TCP, choisis aléatoirement. Grâce à des chiffrements successifs [42], tel un oignon, aucun de ces 3 intermédiaires ne connaît simultanément :

— qui est le client
— qui est le serveur
— quelles sont les données échangées [43]

Le serveur de destination ne sait pas qui est le client. De plus, les services cachés (**onion**) permettent d'anonymiser également le serveur : on ne quitte même plus le réseau tor.

tor offre une bien meilleure *non traçabilité* que l'Internet classique, au prix d'une performance bien moindre. Des attaques restent toutefois possibles (corrélation statistique, couches supérieures trop bavardes, etc)

Signalons toutefois le grand risque d'exploiter un nœud de sortie tor, en première ligne des investigations judiciaires : ils sont en général exploités par des associations ou des entreprises.

3.3.4 Next Generation Networks (NGN)

Les opérateurs de télécommunications, représentées par les organismes de standardisation comme l'ITU ou l'OSI n'ont pas forcément les mêmes intérêts commerciaux ou besoins que les opérateurs de réseaux Internet (voir section 3.1.1.3 en page 51), rien que par le type d'information transportée ou les modèles de facturation, qui peuvent s'opposer à la **neutralité du réseau**.

C'est pour cette raison que les protocoles de télécommunications classiques ont longtemps évolué parallèlement aux protocoles IP. Aujourd'hui, il devient de plus en plus coûteux pour les opérateurs de télécommunications historiques – devenus entre temps entre autre des fournisseurs d'accès Internet (**FAI**) en d'offrant simultanément des services **quadruple-play** (télévision,

40. quelques éléments concernant le calcul du checksum en couche 4, par exemple
41. la couche 4 TCP et la couche 3 y sont fusionnées : il est toutefois possible de mettre en place une couche 3 IPv6 complète via **onioncat**, à la performance douteuse
42. le 1er relais (d'entrée) connaît le client et peut déchiffrer l'adresse du deuxième relais fournie par lui et rechiffre à son intention ; le deuxième relais ne connaît pas le client et peut déchiffrer l'adresse du dernier relais ; le dernier relais ou nœud de sortie peut déchiffrer l'adresse du serveur et le contenu
43. si du chiffrement de bout en plus est en plus utilisé, p.ex. HTTPS, elles ne seront connues que du client et du serveur

téléphone fixe et mobile, Internet) – de développer et d'exploiter deux infrastructures parallèles et de ne pouvoir facilement intégrer les produits proposés à leurs clients.

De plus, les réseaux IP généraux n'offrent pas les garanties suffisantes aux opérateurs télécoms classiques, notamment en ce qui concerne :

— la sécurité
— la facturation
— la qualité de service
— la mobilité (roaming)

Même s'il est possible au sein d'un réseau restreint d'opérateur (en particulier avec un réseau core **MPLS**) d'assurer ces besoins via des outils développés en interne et du personnel qualifié, le coût de maintenance peut devenir prohibitif, et l'interaction entre plusieurs opérateurs rendue difficile. Enfin, les anciennes technologies (GSM 3G, téléphonie **SS7**[44] analogique ou ISDN) ne seront que difficilement intégrables, car elles n'utilisent pas le réseau core **MPLS**.

La vraie réponse est une intégration complète et standardisée de ces deux mondes : en effet, les grands réseaux IP sont en général intégrés et transportés par des technologies hybrides couche 2/3 (réseau **core**), comme p.ex. **MPLS** (un standard **IETF** qui dérive à la fois des techniques d'ATM et des résultats du monde IP) ; l'accès à la télévision par **VDSL** se base sur des standards IP et la téléphonie IP pourra remplacer à terme la téléphonie **SS7** analogique ou ISDN qui n'est plus développée.

Les *Next Generation Networks* (NGN) sont une évolution d'architecture de télécommunications visant à cette intégration. Leurs caractéristiques principales sont :

— transporte toutes les données et services sous forme paquet
— sont construits sur la pile de protocole IP (**all-IP**) – dans la plupart des cas
— touche les réseaux **core** et les réseaux **d'accès**
— interface les réseaux historiques (téléphonie SS7 analogique et ISDN, GSM) via des passerelles (**gateway**)
— utilise toutes les technologies d'accès possibles
— met en place de la qualité de service (**QoS**) via des contrats (**SLA**) garantis par l'infrastructure
— offre des services de mobilité (**roaming**)
— offre des plateformes d'identification et de facturation généralisées
— ajoute des services de chiffrement et d'authentification, tout en respectant les lois locales concernant les services de surveillance des télécommunications (pas de chiffrement de bout en bout des services et données de base du réseau) et assurant une confidentialité complète (données et entêtes) par *tronçon*.

Les NGN capitalisent sur de nombreux protocoles existants d'Internet comme IP, TCP, UDP, SIP, H.323, **RADIUS**/**Diameter**[45], **IPsec** ainsi que des technologies comme l'isolation (**firewall**, notion de zones restreintes, etc). Il s'y ajouteront des services d'activation multimédia (par exemple sous forme de services Web) et de passerelles entre technologies et des protocoles propres de signalisation NGN-IMS[46].

Le déploiement des NGN se fait en deux phases :

1. **NGN light** ou partiel : le NGN est déployé au sein d'un ou plusieurs opérateurs, mais les échanges entre opérateurs sont toujours effectués à l'aide de protocoles classiques

44. encore utilisé aujourd'hui pour la signalisation de l'interconnexion téléphonique entre opérateurs, peu sécurisé même lorsque transporté sur IP
45. protocoles **AAA**, *Authentication, Authorization, Accounting/Auditing* : **authentification**, **contrôle d'accès** et **traçabilité**
46. remplaçant SS7

comme **SS7**

2. **NGN full** : au niveau international, de plus en plus d'opérateurs exploitent uniquement les protocoles all-IP au sein des NGN.

Ce n'est qu'en déploiement complet que les avantages des NGN seront perceptibles, soit la qualité de service négociable de bout en bout sur Internet, la mobilité complète de couche 3 et le chiffrement par tronçon (assurant confidentialité par défaut et respect des lois nationales de la surveillance des télécommunications).

L'architecture Internet futuriste **SCION**[47] veut permettre un déploiement partiel entre systèmes autonomes (**AS**) participants, assurant une résistance aux pannes, la sécurité et la performance sur les NGNs, même en présence d'agents malveillants dans les équipements et chez les opérateurs. Elle fait partie des projets exploratoires de l'IETF dans le domaine des réseaux *path-aware* et est déjà en test dans des institutions suisses.

Le triangle fournisseurs de contenu et de services (notamment les GAFAM), utilisateur et opérateurs télécoms modélise bien des intérêts divergents : la **neutralité du réseau** est centrale pour les fournisseurs de contenu et de services, la sécurité et la sphère privée essentielle pour les utilisateurs et les services offerts et leur facturation pour les opérateurs. Ce sont les interactions au sein de ce triangle qui décideront si et quand les NGN seront réellement exploités en NGN full.

3.4 Sécurité du réseau

3.4.1 Introduction

Par défaut, IPv4 n'incorpore pas de mécanisme de sécurité, que cela soit pour :

l'intégrité : un simple checksum d'entête : pas de signature numérique (voir section 6.6.3 en page 107) qui garantirait l'authenticité des messages

la confidentialité : tout est en clair par défaut

la disponibilité : pas de mécanisme de base

Les couches supérieures peuvent authentifier et chiffrer : mais les méta-données de couche 2, 3 et 4 ne sont pas protégées. Il existe toutefois des extensions pour la sécurité en couche 3 : citons par exemple les **VPN**s, comme le protocole **IPsec**, qui permettent d'assurer, dans une certaine mesure, l'intégrité des messages (méta-données et données), et la confidentialité des données. IPv6 oblige l'implémentation d'IPsec sans toutefois prévoir de mécanisme de configuration automatique[48]. Une certaine haute disponibilité (**HA**) peut être mise en place par le DNS, les protocoles de routage ou les **CDN**. Un **firewall** (niveau réseau ou applicatif, comme un **WAF**, *Web Application Firewall*) et de la détection d'intrusion (**IDS**) peuvent renforcer la sécurité.

3.4.2 Sécurité en couche 2

Certains switches peuvent être configurés pour appliquer quelques éléments de sécurisation de couche 2. Ils peuvent dans certains cas même inspecter les entêtes des couches supérieures pour bloquer certaines attaques. Parmi les fonctions de sécurité des switches déployés en entreprise, citons :

47. *Scalability, Control and Isolation On NGNs*, https://www.scion-architecture.net/
48. concept de cryptographie opportuniste, voir RFC-7435 et le projet FreeS/WAN – non déployé

— des limitations pour chaque port où se trouvent uniquement des machines et non des liaisons avec d'autres switches : maximum d'adresses MAC, de trafic multicast ou broadcast, de nombre de requêtes du protocole ARP ou de protocoles de couches supérieures (p.ex. DHCP) ; blocages liés au protocole spanning tree [49], aux VLAN ou à des protocoles propriétaires Cisco (DTP, CDP)
— de l'anti-spoofing du protocole ARP (Dynamic ARP inspection) ou IP (IP Source Guard)
— des blocages contre des réponses de protocole réservées à des serveurs (DHCP Snooping, filtrage LLDP/NDP [50])
— de l'authentification pour l'accès à un réseau chiffré ou l'attribution à un **VLAN** (**802.1x**, **EAP**) en WiFi, voire Ethernet

3.4.3 Compatibilité des besoins de sécurité et de surveillance

Le tableau ci-dessous résume les moyens techniques de protection des méta-données (entête IP : adresses IP ; entêtes de couche 4 : numéros de ports) et des données suivant les protocoles utilisés et la compatibilité avec les besoins de surveillance des autorités légitimes [51] :

techniques	sécurisation		LSCPT		performance
	méta	données	méta	données	
HTTP			✓	✓	✓
HTTPS		✓	✓		✓
tor + HTTP	✓	(✓)			
tor + HTTPS	✓	✓			
VPN + HTTPS	(✓)	✓			(✓)
NGN full (voir page 79)	✓	✓	✓	✓	✓

HTTPS chiffre le payload après la couche 4 (les entêtes de couche 4 et inférieures restent en clair) : les méta-données de couche 3 (adresse IP) ne peuvent pas être cachées car elles sont nécessaires pour le routage IP. On peut les cacher dans une certaine mesure et au prix de délais plus ou moins importants par des VPNs ou du routage par onion (**tor**).

En HTTPS, si l'on connaît l'adresse IP et le port du serveur consulté, on peut facilement télécharger le certificat X.509 TLS/SSL associé et donc le (ou les aliases) du nom de domaine accédé, car originellement un seul certificat était possible par adresse IP et port [52]. Le problème est bien sûr agravé si un DNS en clair [53], même avec les extensions d'intégrité **DNSSEC**, est utilisé.

Les **NGN**-full (voir section 3.3.4 en page 78) prévoient un chiffrement par tronçon (VPNs tunnelisés par IPsec), protégeant les données *et* méta-données, compatible, en déploiement complet *et* international des NGN, avec les besoins des autorités de surveillance, dans la mesure où du chiffrement de bout en bout n'est *pas* utilisé en plus. Pour le moment, le déploiement de cette solution n'est pas planifiée.

49. aussi pour protéger le réseau contre des boucles
50. Link Layer / Network Discovery Protocol, utilisé en IPv6
51. en Suisse : Loi fédérale sur la surveillance de la correspondance par poste et télécommunication (LSCPT) et loi sur le service de renseignement (LRens)
52. l'extension **SNI** – *Server Name Indication* – qui permet d'héberger plusieurs certificats sur la même adresse IP, transmet *en clair* le nom du domaine désiré : seul **ESNI**, très récent, améliore la confidentialité sur quel site est consulté et seulement si l'adresse IP est servie par un reverse-proxy qui sert énormément de domaines différents (gros hébergeurs ou **CDN** – *Content Distribution Network* comme par exemple CloudFlare) !
53. des extensions pour DNS sur TLS (**DoT**) ou HTTPS (**DoH**) existent qui limitent les risques de surveillance par les autorités ou le fournisseur d'accès Internet : elles posent toutefois des problèmes aux DNS locaux et simplifient la collecte de métadonnées par le fournisseur de ce service

3.5 Exemple de réseau complexe

Un réseau complexe (un internet, relié ou non à Internet) va unifier de nombreuses technologies de couche 2 au sein d'une même couche 3, éventuellement plusieurs couche 3 en présence de **gateway** de couche 7 pour accéder à des réseaux non IP (ici un exemple **LoRa**, où Internet est utilisé ensuite pour intégrer via **LoRaWAN**) ou de **NAT**.

Des réseaux privés virtuels (**VPN**) permettent de construire des topologies logiques en couche 2 ou 3, comme par exemple relier deux réseaux entre eux via Internet, même de fournisseurs différents, ou des travailleurs en déplacement (**road warrior**).

La sécurité est présente sous la forme de confinement (par exemple ici segmentation, au sein de l'entreprise entre un réseau interne et un réseau **DMZ** par **firewall**).

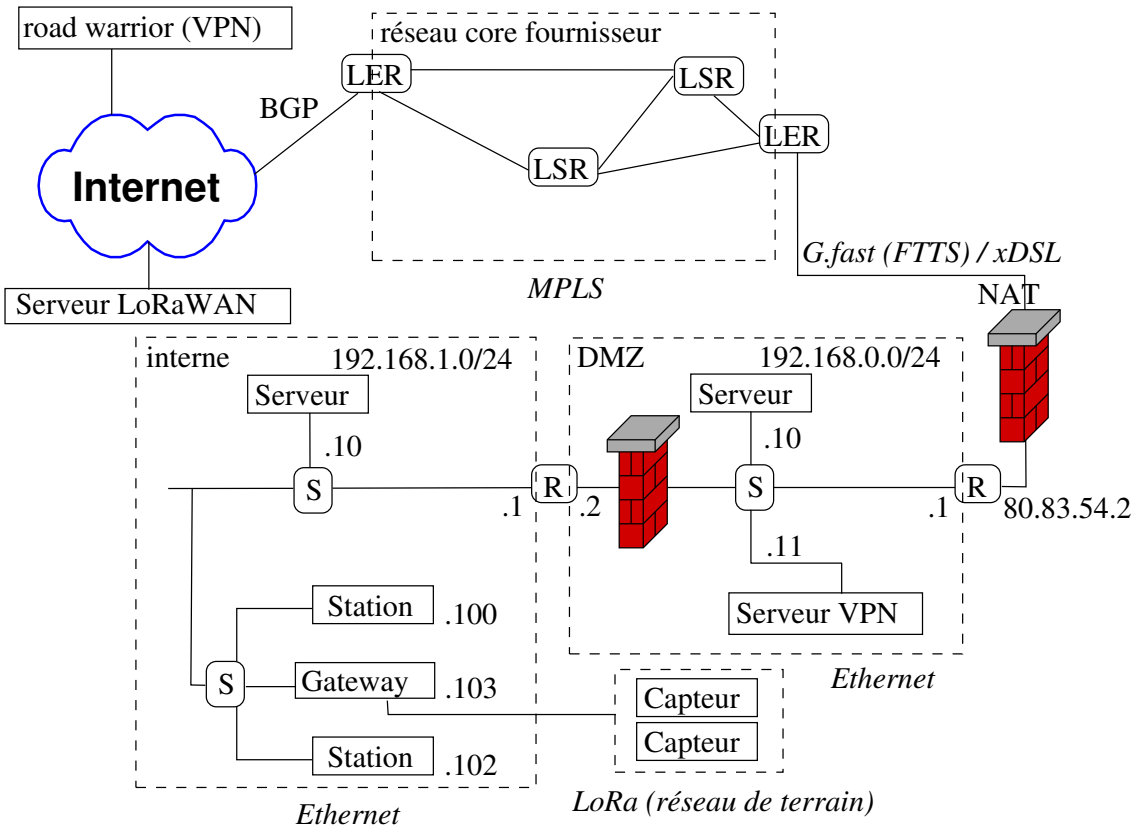


FIGURE 3.3 – Exemple de réseau complexe

Sur le schéma, on voit un routeur de bord Internet chez le fournisseur, qui utilise le protocole BGP pour annoncer des routes sur Internet – c'est possible dès lors que le fournisseur, voire même une entreprise, est composé d'au moins un système autonome (AS, voir section 3.2.3.3.2 en page 70) et dispose de ses propres sous-réseaux indépendants du fournisseur (**plage PI**, voir section 3.1.3.5 en page 55), d'au moins une taille /24 en IPv4. On y voit aussi un réseau **core MPLS**, (voir section 3.2.3.4 en page 71).

Chapitre 4

Couche transport (4)

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison
1	Physique

Sommaire

Le but de ce chapitre est de présenter le rôle de la couche transport (en anglais *transport layer*), la couche qui gère les communications de bout en bout entre **processus** – ou applications – à travers la couche réseau et en fournissant des services **fiables** ou non, de type **connecté** ou non, de manière fonctionnelle puis sur des exemples pratiques.

4.1 Rôles

4.1.1 Complémentarité avec la couche liaison

Comme la couche liaison, la couche transport peut implémenter un **protocole fiable**[1] : la différence entre ces deux couches est liée à la notion de **portée** (voir section 0.2.3.6 en page 16) : en effet, si l'ensemble des couches 2 formant un grand WAN de couche 3 implémentent les mécanismes des protocoles fiables (gestion des erreurs, établissement, contrôle de flux...), elles n'assurent pas ces mécanismes de bout en bout à travers tout le **WAN**.

Par exemple, les deux partenaires de bout en bout peuvent ne pas pouvoir traiter les données de l'autre direction assez rapidement (surcharge), ou la congestion d'un routeur peut mener à la perte de datagrammes. Il faut savoir que la **congestion** est aujourd'hui la raison principale de pertes de datagrammes dans un WAN comme Internet : bien avant les erreurs de transmission !

Enfin, il est à noter que certaines couches 2 n'implémentent pas l'ensemble des fonctions attendues pour un protocole fiable (par exemple, Ethernet – sans sous-couche **LLC** dans le modèle IP implémente uniquement la détection d'erreur et non pas la retransmission[2]). Si l'application le *demande*, la couche 4 implémentera les fonctions manquantes.

La couche liaison et la couche transport sont donc *complémentaires*, chacune dans son domaine de portée respectif.

4.1.2 Fonctions de la couche transport

On peut classer les fonctions de la couche transport dans les catégories suivantes :

accès à des services connus : si l'adresse de couche 3 identifie une machine, l'adresse de couche 4 identifiera le service (une application), par exemple en TCP ou UDP sur IP par le concept de (numéro de) **port** : en général un des ports, situé sur le terminal serveur est un service connu, et l'autre, situé sur le terminal client, est un numéro quelconque (port fantôme, temporaire)

identification des flux : le **nexus** identifie un flux UDP ou une connexion TCP de manière *unique* : il est composé de quatre valeurs : le **quadruplet**[3] formé des deux adresses IP des partenaires, et leurs deux numéros de **port** ; les flux de couche 4 sont donc multiplexés en couche 3 sur le premier terminal partenaire, et le nexus permet le démultiplexage sur le second terminal partenaire et vice-versa

protocole fiable : en cas de besoin, l'application peut choisir un protocole de couche 4 fiable, de bout en bout

Le choix du protocole de couche 4 dépend en effet des besoins de l'application : par exemple, les protocoles multimédia interactifs ne désirent pas de retransmission en cas d'erreur ou de pertes de données, mais un délai le plus faible possible : dans la pile IP, on utilisera alors plutôt le protocole de couche 4 UDP. A contrario, les applications informatiques nécessiteront une implémentation complète : TCP assurant la présence du partenaire (connexion) et le protocole fiable.

Dans la section suivante, nous allons tout d'abord étudier la variante *luxueuse* (TCP), puis ensuite la variante simplifiée (UDP).

1. on parle parfois de protocole *sûr*, mais pas dans le sens du chiffrement ou de l'authentification
2. sinon la couche 2 violerait les besoins de certaines applications
3. ou **quintuplet**, en tenant compte également du fait que les espaces TCP et UDP sont séparés

4.2 TCP – Transmission Control Protocol

4.2.1 Rôles

Le but du protocole TCP, orienté **connexion**, est d'offrir aux applications qui en ont *besoin* l'assurance que tous les datagrammes ont été délivrés correctement à leur destinataire (**protocole fiable**). Par exemple, les applications informatiques (transfert de fichier, transaction de base de données[4]) sont généralement concernées

TCP propose les fonctions suivantes :

- multiplexage en couche 3 de plusieurs flux TCP couche 4 (démultiplexage grâce au nexus unique)
- gestion de la connexion (phase d'ouverture et de fermeture, négociations d'options éventuelles, fermeture brutale...)
- découpage en segments des données des couches supérieures
- envoi de données bidirectionnel simultané possible (avec optimisation **piggy-backing**[5])
- protocole à fenêtre s'adaptant aux caractéristiques WAN observées en temps réel
- fiabilité (quittance de bonne réception de la part du récepteur, assurance de l'ordre des données par numéros de séquence et réordonnancement éventuel, suppression des doublons éventuels)
- contrôle de flux (ne pas surcharger le récepteur)
- options récentes (p.ex. réaction aux congestions sur les routeurs, optimisations liées aux nouvelles performances des WAN, etc) – qui peuvent parfois poser des problèmes de compatibilité

En fait, TCP reçoit des données d'un protocole de niveau supérieur dans un mode orienté **stream**, c'est-à-dire caractère par caractère. Les octets sont alors groupés dans des segments TCP avant d'être transmis au protocole IP. L'impact direct sur l'application, sans utilisation d'une couche d'abstraction supplémentaire, est l'imprévisibilité du découpage des données : vous demandez 500 octets, mais vous recevez 240.

TCP est un **protocole client/serveur**, avec un serveur en attente d'une connexion et un client qui ouvre la connexion.

4.2.2 Ports

Pour que deux applications puissent communiquer ensemble via TCP, une d'entre elles doit être configurée en mode serveur et l'autre en mode client. Les protocoles qui font appel à TCP utilisent un numéro de port dont la concaténation avec l'adresse IP locale adéquate forme l'adresse d'un **socket** ou prise réseau.

Sous UNIX, le socket peut être associé à un fichier, que l'on peut accéder normalement. Les appels systèmes usuels `read(2)`, `write(2)` et `close(2)` peuvent être utilisés de la même manière que sur des périphériques de type caractère (**character device**) comme des lignes série ou terminaux virtuels[6]. En plus, l'appel système `shutdown(2)` peut être utilisé pour terminer le transfert de données dans une direction (voir section 4.2.6 en page 88).

4. encore que TCP n'est pas forcément optimal en raison du coût d'ouverture et de fermeture potentiellement à chaque transaction : des remplacements sont en cours d'étude, comme par exemple **T/TCP**.

5. les confirmations d'une direction voyagent avec les données de l'autre direction

6. en pratique cela signifie que read(2) et write(2) ne lisent, respectivement n'écrivent, pas forcément toutes les données attendues – une boucle, effectuée notamment par les fonctions de la bibliothèque C standard fread(3), fwrite(3) et fprintf(3) et fscanf(3) est nécessaire.

En général, le port du côté serveur est un port connu (**well-known port**, voir section 4.2.3 en page 86), et celui du client un port alloué dynamiquement [7]. Sous systèmes POSIX (UNIX par exemple), les ports en-dessous de 1024 sont réservés à root, dans le but d'éviter de l'interception de mots de passe par exemple.

4.2.3 Services connus

En plus des ports alloués dynamiquement du côté du client, il est nécessaire de définir certains ports liés à des services TCP bien connus [8], pour que les clients puissent s'y connecter sans besoin d'un annuaire dynamique.

port	service	description
20	FTP-Data	File Transfer (RFC-959)
21	FTP-Control	
22	SSH	connexions interactives sécurisées, exécution de commandes distantes et transfert de fichier
23	TELNET	connexion interactive non sécurisée (RFC-854/859)
25	SMTP	Simple Mail Transfer (RFC-822/821)
37	NTP*	Network Time Protocol (RFC-1119)
43	WHOIS	interrogation des bases de données réseau WHOIS
53	DNS*	Domain Name Service
80	HTTP	Hyper Text Transport Protocol (WWW)
119	NNTP	News Network Transport Protocol (Usenet news)
110	POP3	Post Office Protocol (consultation et transfert d'une boîte-aux-lettres)
137-139, 445	NETBIOS*	NetBIOS Name/Datagram/Session service – partage de fichier Microsoft
220	IMAP3	IMAP3 (consultation de boîtes-aux-lettres)
443	HTTPS	HTTP sur SSL
515	PRINTER	Impression via protocole LPR
587	SUBMISSION	Mail Submission Protocol (RFC-4409)
631	IPP	Internet Printing Protocol (CUPS)
993	IMAPS	IMAP sur SSL
995	POP3S	POP3 sur SSL
6000	X11	1er écran graphique X11
6667	IRC	Internet Relay Chat

Les entrées avec une étoile sont aussi exploitées en UDP. Par exemple, le **DNS** utilise TCP uniquement en cas de nécessité (p.ex. réponse de grande taille).

FIGURE 4.1 – Quelques services connus TCP

7. originellement séquentiellement dès 1024 ; aujourd'hui aléatoirement pour des raisons de sécurité.

8. voir votre fichier `/etc/services`, sur votre machine UNIX, ou http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml ou encore la figure 4.1 en page 86 ci-après

4.2.4 Format du datagramme TCP

Chaque segment TCP est formé d'une zone d'entête et d'une zone de données placées à la suite de l'entête IP (en fait dans la zone des données IP), chaque ligne représentant 32 bits et les valeurs entre parenthèses représentant le nombre de bits de chaque champ :

<table>
<tr><td colspan="8">Source Port (16)</td><td colspan="2">Destination Port (16)</td></tr>
<tr><td colspan="10">Sequence Number (32)</td></tr>
<tr><td colspan="10">Acknowledgment Number (32) – valide si et seulement si flag ACK</td></tr>
<tr><td>Offset (4)</td><td colspan="2">Réservé (6)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window (16)</td></tr>
<tr><td colspan="8">Checksum (16)</td><td colspan="2">Urgent pointer (16) – ssi URG</td></tr>
<tr><td colspan="10">Options éventuelles et bourrage – ssi Data Offset $>$ 5</td></tr>
<tr><td colspan="10">Données éventuelles des couches supérieures</td></tr>
</table>

FIGURE 4.2 – Format du datagramme TCP

Après les numéros de port, le champ de numéro de séquence contient la position des données dans le flux de caractères transmis par l'émetteur. Le champ de numéro de quittance contient la valeur du prochain numéro de séquence attendu.

Le champ de décalage de données (Offset) spécifie la longueur de l'entête TCP en mots de 32 bits, c'est-à-dire généralement 5 mots de 32 bits (20 octets).

Le champ réservé contient aujourd'hui 3 bits réservés et 3 drapeaux (NS, CWR et ECE dont le rôle est l'implémentation optionnelle couche 4 de l'extension **ECN** – gestion de la **congestion**).

Les 6 champs suivants ne comptent qu'un seul bit et sont donc appelés des drapeaux (flags) : URGent (des données urgentes pointées dans l'entête sont à remettre à l'application), ACKnowledgment (confirmation de données de l'autre direction), PuSH function (envoi immédiat à la couche supérieure), ReSeT (arrêt catastrophique de l'échange ou service non écouté à l'ouverture), SYNchronize (synchronisation des pointeurs – numéros de séquence initiaux, lors des 2 premières phase sur 3 de l'ouverture de connexion), FINish transmission (fermeture unidirectionnelle puis bidirectionnelle).

Le champ fenêtre est une valeur indiquant le nombre d'octets que le récepteur est disposé à accepter (elle sert au **contrôle de flux** et de **congestion**).

Le champ de checksum représente le complément à 1 de la somme de tous les mots de 16 bits du segment TCP. Ce code détecteur n'a pas la puissance d'un CRC – que l'on espère utilisé dans les différentes couches 2 traversées – mais permet toutefois de détecter des problèmes simples.

Enfin, si des options sont présentes, elles sont alignées (par **bourrage**) au prochain multiple de 32 bits. Citons par exemple les options permettant un horodatage (**timestamp**) pour la gestion des variations de délai ou des paramètres du protocole fiable à fenêtre pour l'adapter aux WAN modernes.

4.2.5 Établissement de la connexion

L'établissement de la connexion se fait par l'échange de trois messages (ouverture en 3 phases). C'est le client qui initie l'ouverture (**ouverture active**). Les partenaires synchronisent les numéros de séquence[9] et établissent la taille initiale de la fenêtre.

9. si les numéros de séquence initiaux sont suffisamment aléatoires, cela est aussi une protection contre une attaque de type **half-open**.

1. premier message (du client) avec le bit SYN positionné et la valeur initiale du numéro de séquence du premier partenaire
2. deuxième message (réponse du serveur au premier) avec les bits SYN et ACK positionnés et la valeur initiale du numéro de séquence du second partenaire. Le bit ACK confirme la réception du premier numéro de séquence.
3. troisième message (réponse du client au deuxième) avec le bit ACK positionné pour confirmer la réception

Le quadruplet (adresse IP source, port source, adresse IP destination, port destination) forme l'*identifiant* (unique) (appelé **nexus**) de la connexion.

4.2.6 Fermeture de la connexion

La fermeture de la connexion fait appel à quatre (voire trois) messages. Les flux dans les deux directions peuvent être coupés indépendamment. La fermeture peut être initiée par chacun des partenaires. Elle de type **gracefull close** car elle assure qu'aucune donnée n'est perdue.

1. premier message avec les bits FIN et ACK positionnés.
2. deuxième message (réponse au premier) avec le bit ACK positionné. Le flux est ainsi coupé dans un sens uniquement. L'émetteur du premier message ne peut plus émettre de messages contenant des données mais la réception de toutes ses données est confirmée
3. troisième message avec les bits FIN et ACK positionné (en sens inverse du premier). Ce message ne doit pas impérativement suivre le deuxième : des messages de données provenant de l'émetteur du deuxième message et leurs confirmations peuvent encore le précéder.
4. quatrième message (réponse au troisième) avec le bit ACK positionné. Le flux est ainsi coupé dans l'autre sens mais la réception de toutes les données est confirmée.

Les phases 2 et 3 peuvent être combinées pour une fermeture simultanée des deux directions.

4.2.7 Gestion de la fenêtre

Les données sont envoyées au destinataires, encapsulées dans des datagrammes TCP. Plusieurs datagrammes peuvent circuler dans le réseau à la suite sans attendre de confirmation : la taille de fenêtre indique le maximum d'octets qui peuvent circuler dans le réseau sans confirmation. Dès qu'une confirmation est reçue, la fenêtre autorisée est décalée d'autant.

Le destinataire d'un datagramme de données va déposer ces données dans un tampon à destination de la couche supérieure [10], et les données seront au final transmises à la couche supérieure (avec éventuel réordonnancement). Il va confirmer les données reçues par un datagramme en retour avec le bit ACK positionné et le numéro de confirmation adapté.

Il se peut toutefois que la place libre dans ce tampon devienne plus petite que la taille de fenêtre actuelle. Dans ce cas, le destinataire va profiter des confirmations qu'il envoie en retour pour adapter la taille de fenêtre de l'émetteur (champ *window*), et donc contrôler le flux de l'émetteur.

Il faut noter que ce changement peut dans certains cas amener à des problèmes de performance ou des données jetées à la réception (les données en transit peuvent être trop volumineuses pour le tampon de réception, et l'émetteur peut devoir découper de manière inefficace).

10. sauf en cas de duplicats, ou de données complètement hors séquence

Les confirmations peuvent être groupées. De plus, les confirmations sont envoyées avec les données de l'autre direction (efficacité meilleure, **piggy-backing**). S'il n'y a pas de données à émettre, le numéro de séquence d'envoi ne change pas.

La taille de fenêtre réellement utilisée par un émetteur dans une direction donnée est toutefois limitée par le protocole **slow start** [11] qui permet à TCP de s'adapter au débit disponible.

4.2.8 Gestion de la minuterie de confirmation

TCP fonctionnait initialement sur les principes **CONTINUOUS REQUEST GO BACK-N** [12] en mode de retransmission implicite.

GO BACK-N implicite signifie qu'en cas de perte d'un message, les retransmissions se font depuis le point de la première erreur [13], et qu'elles n'ont lieu que lorsque la minuterie (**timer**) d'un message est échue. Or le délai entre l'émission d'un message et la réception de la quittance correspondante peut fluctuer énormément (quelques millisecondes dans un LAN, quelques secondes dans Internet).

Ce délai peut même varier fortement au cours d'une connexion. Si le timer est trop court, des retransmissions inutiles ont lieu. S'il est trop grand et s'il y a eu une erreur, elle ne sera détectée que trop tard : un grand volume de données doit être réémis (**GO BACK-N**) et le flux subit un *à coup*. C'est pourquoi ce délai doit être ajusté en permanence. TCP mesure ce délai (**RTT**) régulièrement et calcule une moyenne coulissante ainsi que la déviation standard. Avant l'émission d'un nouveau message un timer est calculé sur la base de ces deux valeurs.

4.2.9 Gestion des congestions

Le mécanisme de contrôle du flux (fenêtre) ne fonctionne que de bout en bout. Il protège contre une surcharge du destinataire mais pas contre une surcharge du réseau (routeur, ligne...). Une telle surcharge est appelée **congestion**. Lorsque leurs tampons sont pleins [14] les routeurs détruisent des messages. Ils peuvent informer la source par un message ICMP mais il n'y a pas de garantie que de tels messages arrivent. La couche IP du destinataire ne remarque en général pas la disparition d'un message. C'est la couche TCP émettrice qui constate l'absence de quittance pour un de ses messages et qui réémet (**GO BACK-N**) ce qui accentue la surcharge du réseau !

Pour éviter ce problème un émetteur TCP doit réduire spontanément la taille de sa fenêtre d'émission lorsque une confirmation manque après le délai prévu. Il la réduit rapidement et recommence prudemment [15] à la faire croître afin d'éviter des phénomènes d'oscillation. La taille de la fenêtre ayant provoqué une congestion est mémorisée et la croissance de la taille de la fenêtre est très ralentie à l'approche de cette valeur.

11. TCP essaie d'occuper au mieux le débit disponible, en commençant par un débit faible, grâce à une taille de fenêtre effective petite, augmentant rapidement : en cas de perte un algorithme spécial est activé, voir section 4.2.9 en page 89
12. il existe des options TCP, aujourd'hui largement supportées, pour implémenter le **CONTINUOUS REQUEST SELECTIVE REPEAT**, parmi d'autres extensions du protocole TCP.
13. en retransmettant éventuellement des messages pourtant bien reçus
14. l'extension **ECN** permet d'informer l'émetteur avant que cela n'arrive, voir section 4.2.11.1 en page 90
15. à l'ouverture de la connexion, l'augmentation est bien plus rapide.

4.2.10 Algorithme interactif

De manière à augmenter l'efficacité des transactions interactives (notamment **TELNET**), TCP peut activer un algorithme de groupement de caractères par accumulation dans un tampon (introduisant un petit délai supplémentaire), nommé **NAGLE**. Ce comportement est contre-productif pour d'autres applications et doit être désactivé par l'application.

4.2.11 Améliorations et alternatives à TCP

4.2.11.1 Améliorations de TCP

TCP a été conçu il y a très longtemps : étonnamment, ce protocole a bien vieilli – notamment car de nombreuses extensions corrigeant certains des problèmes découverts ont été ajoutées, en général de manière rétro-compatible par une négociation d'options à l'ouverture de connexion ou l'utilisation de champs réservés, ce qui a parfois causé des problèmes à certains firewalls, p.ex. pour **ECN**.

Parmi ces extensions, citons celles liées à la performance (confirmation sélective – protocole de type **SELECTIVE REPEAT** plutôt que **GO BACK-N** ne retransmettant que les messages en erreur, facteur d'échelle de la taille de fenêtre (*window scaling*), améliorations des protocoles de réémission) ou au traitement de la congestion sur LAN (**ECN**, *Enhanced Congestion Notification*).

4.2.11.2 Alternatives à TCP

Les améliorations ci-dessus sont très appréciables dans le contexte de l'Internet moderne : toutefois, des variantes plus spécifiques existent : citons **T/TCP**, destiné au transactionnel ou **DCTCP**, *Data Center TCP*, pour les centres de données, ou enfin le projet de recherche européen **MPTCP** (*Multipath TCP*), découplant IP et TCP.

Un développement complètement différent est le protocole **SCTP** (*Stream Control Transmission Protocol*), qui au-dessus d'IP (ou, éventuellement d'UDP, voire TCP) implémente un mélange de fonctions de TCP et UDP et s'adapte mieux aux fonctions avancées d'un réseau IP comme la haute fiabilité, le choix du routage, la résistance à certaines attaques et une détection d'erreur améliorée.

Signalons aussi que Google teste actuellement dans son navigateur Chrome le protocole fiable **QUIC**, implémenté au-dessus d'UDP et qui offre de nombreuses fonctionnalités intéressantes par exemple pour la minimisation du délai, le chiffrement, le multiplexage, etc. Sa caractéristique principale est de modéliser le *tuyau WAN* entre les deux partenaires et d'adapter son débit dans le but de limiter l'usage des queues des routeurs et donc d'optimiser le délai et d'éviter la **congestion**.

Quant aux anciens protocoles OSI de couche 4, comme par exemple **TP4**, du standard OSI **X.224**, leur utilisation est très anecdotique : citons par exemple le protocole de couche 7 **RDP** (*Remote Desktop Protocol*) qui est utilisé dans l'environnement Microsoft pour la connexion graphique distante [16], et qui est basé sur TP4, probablement car RDP s'inspire du protocole de session OSI T.128. Aujourd'hui, cet ensemble RDP sur TP4 est transporté sur TCP (port 3389 par défaut) et donc fait double emploi.

16. similairement à VNC ; sous UNIX, le standard X11 et VNC sont utilisés dans ce but, les deux protocoles sont aussi basés TCP

4.3 UDP – User Datagram Protocol

Lorsque les services du protocole TCP ne sont pas nécessaires, ou même contre-productifs (p.ex. forte interaction temps réel au sein de protocoles multimédias interactifs, scrutation de nombreux équipements, **broadcast** ou **multicast**, passage de firewalls client à client...), il est possible de se contenter du protocole UDP qui travaille en mode non connecté et non fiable (les datagrammes peuvent changer d'ordre, se perdre [17]). Seul le rôle de (dé-)multiplexeur et donc de **nexus** et la notion de **port** sont identiques à TCP. C'est l'application qui doit se charger de répartir les données dans le ou les datagrammes UDP et gérer leur réordonnancement éventuel à la réception via un entête supplémentaire ad-hoc.

port	**service**
7	echo
37	time
53	DNS
67/68	BOOTP/DHCP
69	TFTP
111	RPC portmapper (RFC-1057) – *Remote Procedure Calls*
161/162	Simple Network Management Protocol (RFC-1157)
250	RIP (Routing Information Protocol, RFC-1058)
517	TALK
525	timed (time sync)

FIGURE 4.3 – Quelques services connus UDP

Chaque segment UDP est formé d'une zone d'entête et d'une zone de données placées à la suite de l'entête IP (en fait dans la zone des données IP). Les unités ci-dessous sont en bits.

Source port (16)	Destination port (16)
Length (16)	Checksum (16)
...(data)	

FIGURE 4.4 – Format du datagramme UDP

Les appels systèmes d'entrées-sorties sont, contrairement à TCP, spécifiques à l'interface **socket** car le mode datagramme n'est pas implémentable sous forme de fichier classique UNIX : citons par exemple `sendto(2)` ou `recv(2)`.

17. ou se corrompre car originellement, les checksums UDP étaient optionnels.

Chapitre 5

Couche session (5)

7	Application
6	Présentation
5	**Session**
4	Transport
3	Réseau
2	Liaison
1	Physique

Sommaire

Le but de ce chapitre est de présenter les rôles de la couche session (en anglais *session layer*) : gérer la sychronisation, le contexte et la reprise de transactions, de manière fonctionnelle puis sur des exemples pratiques.

5.1 Rôles

Les rôles de la couche 5 sont variés : gérer les points de reprises (p.ex. transactions), les jetons, la reconnexion, etc.

Dans le modèle TCP/IP, cette couche est laissée vide, en particulier vu la difficulté d'intégrer certaines fonctions qui pourraient être celles de la couche session dans la hiérarchie stricte des couches du modèle OSI. Par exemple, si l'on considère HTTP comme implémentant un échange client/serveur temporaire (p.ex. l'affichage d'un document) faisant partie d'une session plus large (p.ex. la durée d'une session dans un Webmail), HTTP devrait alors être placé en couche 4 (Hyper Text *Transport* Protocol) alors qu'il est souvent vu en couche 7. De plus, HTTP peut compresser et/ou chiffrer les données et effectue des traitements de jeux de caractères : fonctions qui sont usuellement placées en couche 6.

Le protcole HTTP 2.0 (RFC7540) réorganise les transactions individuelles pour optimiser la performance, similairement à ce que les protocoles SCSI Tagged Command Queuing ou SATA NCQ effectuent dans le domaine du stockage. La spécification de ce protocole échange le placement hiérarchique des couches 5 et 6.

Les **token** d'authentification et d'accès à des services peuvent également être vus comme des implémentations de la couche session.

Pour ces raisons, comme mentionné à la section 0.2.4 en page 18, nous ne décrirons pas la couche session en détail.

Chapitre 6

Couche présentation (6)

7	Application
6	**Présentation**
5	Session
4	Transport
3	Réseau
2	Liaison
1	Physique

Sommaire

Le but de ce chapitre est de présenter le rôle de la couche présentation (en anglais *presentation layer*), la couche chargée du codage des données provenant de la couche application, de manière fonctionnelle puis sur des exemples pratiques.

6.1 Concepts

La représentation des données peut diverger d'un ordinateur ou d'une application à l'autre, par exemple :

- organisation en mémoire et à l'envoi
  - taille des mots-mémoire (8, 16, 32 ou 64 bits)
  - ordre des octets dans un mot-mémoire (**big endian** ou **little endian**, le réseau étant en général big endian (poids fort envoyé en premier), les PC généralement little endian.
  - ordre des bits dans les octets (souvent, le bit de poids faible – *Least Significant Bit*, **LSB** – est envoyé en premier)
- format des données
  - encodage des caractères : ASCII, EBCDIC ; ISO-8859-1 ; UTF-8 et standard universel UNICODE
  - représentation des entiers négatifs en complément à 1 ou à 2
  - représentation nombres en virgule flottante
  - structure de données, sérialisation d'objets et typage structuré (XML, JSON, Protobuf, ASN.1/BER...) – voir aussi section 7.1.3 en page 114
  - encodage de transfert (MIME, base64...), si nécessaire

Le rôle principal de la couche 6 est de convertir la représentation des données entre systèmes utilisant des représentations différentes.

Notons que ces problématiques se retrouvent à chaque fois qu'une couche doit définir le format des **PDU**s, et sont souvent résolus de manière plus simple que la méthode générale d'échange de données présentée dans ce chapitre.

Ajoutons que parmi les rôles de la couche présentation peuvent figurer également la **compression** et le **chiffrement** des données.

6.2 Sémantique et syntaxe

6.2.1 Sémantique des données

Les informations échangées entre les systèmes ont une signification bien précise : un champ numérique peut contenir une température, un âge, une vitesse ou un salaire. Un champ alphabétique peut contenir un nom de personne, un modèle de voiture, etc. Ce sens des données est appelé **sémantique**. Seule la couche 7 du modèle OSI se préoccupe de la sémantique. Pour les autres couches la sémantique des données est indifférente et tous les champs numériques sont traités de la même manière.

6.2.2 Syntaxe des données

La **syntaxe** règle la représentation des données. Par exemple :

- les caractères alphabétiques sont représentés selon **Unicode** et codés dans le jeu de caractère **UTF-8**
- les nombres entiers sont codés en complément à 2, sur 4 octets

Le rôle de la couche 6 est de convertir la syntaxe des données en conservant les valeurs. Dans le contexte de la couche 6, on distingue plusieurs types de syntaxes (voir figure 6.1 en page 97).

la syntaxe locale c'est la **syntaxe concrète** utilisée par la couche 7 d'un des partenaires : elle est fixée par la plateforme matérielle, le système d'exploitation et par le langage de programmation utilisé.

la syntaxe de transfert c'est la **syntaxe concrète** utilisée durant le transfert (sur le réseau). Elle doit être connue des deux instances de la couche 6. Si les syntaxes locales sont identiques alors cette unique syntaxe peut être utilisée également pour le transfert. Si elles sont différentes, on a en général deux conversions : syntaxe locale 1 vers syntaxe de transfert et syntaxe de transfert vers syntaxe locale 2. **BER** (*Basic Encoding Rules*) est un exemple de syntaxe de transfert typée (par l'usage de **tag**s de typage préfixant les données, avec portée du typage selon les besoins).

la syntaxe abstraite décrit la structure des données échangées (l'ordre et le type des champs qui composent un message) sans fixer le codage des différents types de champs. La syntaxe abstraite des messages échangés doit être connue des deux instances de la couche 6 afin qu'elles puissent faire les conversions indiquées ci-dessus. **ASN.1** (*Abstract Syntax Notation One)* est un exemple de langage pour la spécification de syntaxes abstraites.

Le passage par une syntaxe de transfert peut paraître peu optimal. Il permet néanmoins de réduire considérablement le nombre des convertisseurs nécessaires. Pour N syntaxes, il faut dans le pire cas N convertisseurs – de l'ordre de N. Pour une conversion directe de chaque syntaxe dans chaque autre, il faudrait $\frac{N(N-1)}{2}$ convertisseurs – de l'ordre de N^2. Ce passage permet donc d'éviter de multiplier de manière incontrôlable les couches de compatibilité, grâce à l'utilisation d'une syntaxe de transfert commune et donc une **interopérabilité** (voir section 0.1.1.3 en page 3).

6.3 Génération des syntaxes locales et de transfert

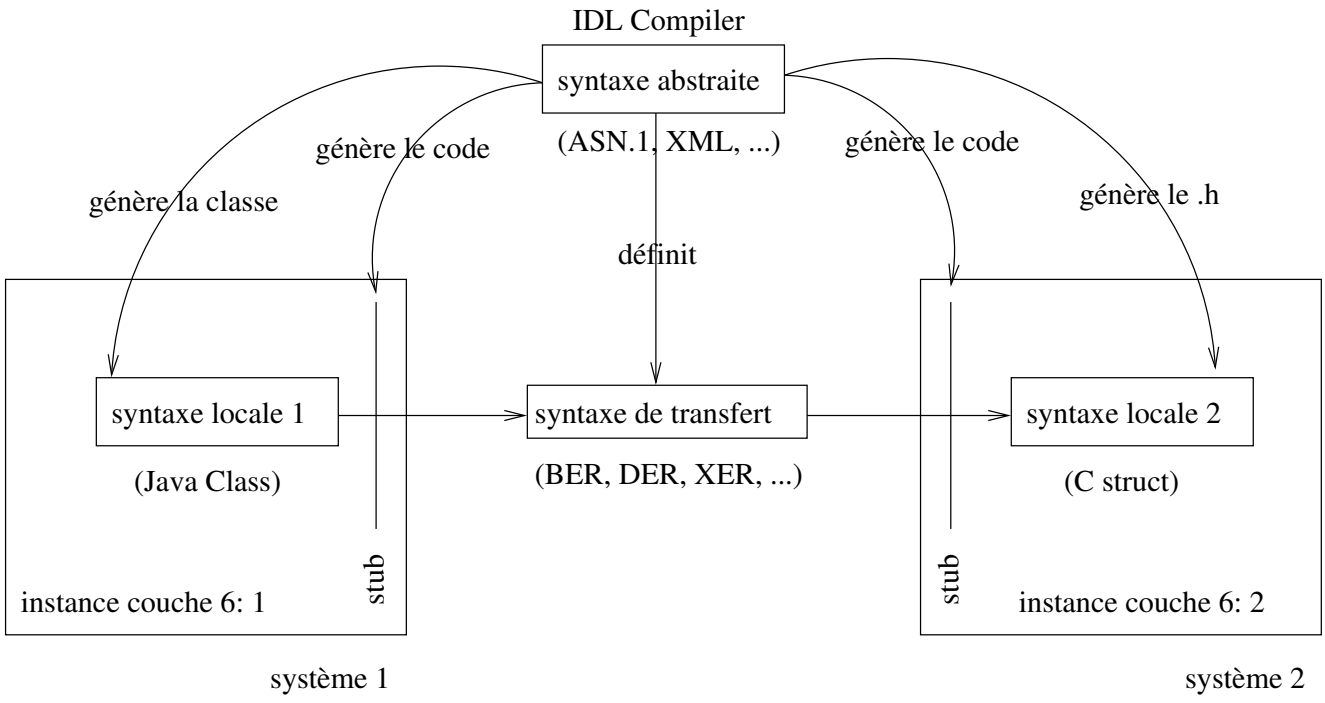


FIGURE 6.1 – Interaction des syntaxes

6.3.1 Principes

Une fois la **syntaxe abstraite** définie (p.ex. décrite en ASN.1), une syntaxe de transfert peut être dérivée automatiquement (p.ex. BER). Ensuite, un compilateur [1] spécifique à chaque langage de programmation peut être utilisé pour générer la **syntaxe locale** et les routines de conversion (**stub** [2]) de la syntaxe locale à la **syntaxe de transfert**.

6.3.2 Exemple simplifié

L'écriture d'une application **socket** TCP sur plateforme PC usuelle montre assez rapidement la nécessité d'une conversion : par exemple, les numéros de port sont à spécifier en syntaxe de transfert de réseau (entier 16 bit non signé, big endian) alors que les PC sont little endian (syntaxe locale). On utilisera alors la fonction `htons(3)` (host to network (unsigned) short) pour convertir de la syntaxe locale 1 à la syntaxe de transfert, puis `ntohs(3)` pour convertir de la syntaxe de transfert à la locale 2 (qui est ici identique, entre 2 PC, à la syntaxe locale 1).

6.3.3 Autres syntaxes de transfert (voire abstraites)

Les réseaux utilisent de manière assez intensive le couple ASN.1/BER (ou des variantes de syntaxe de transfert comme **DER**). Citons par exemple le protocole **SNMP** (*Simple Network Management Protocol*) ou les certificats **X.509** SSL/TLS.

Cependant, les nouvelles applications informatiques liées au réseau sont de plus en plus spécifiées en **XML**, permettant de valider facilement la syntaxe abstraite et la syntaxe de transfert, voire d'utiliser XML également comme syntaxe locale jusqu'à un certain point. XML offre de nombreux avantages, avec une certaine verbosité. Il existe aussi la possibilité d'utiliser une syntaxe abstraite ASN.1 avec une syntaxe de transfert XML (**XER**). Le **JSON** est devenu également très populaire comme syntaxe de transfert.

6.4 Encodage des chaînes de caractères

6.4.1 Introduction

L'encodage des chaînes de caractères est un sujet vaste et fort complexe. Il s'agit de représenter les symboles des langues de manière efficace.

Les approches prises par les premiers jeux de caractères (alphabet télégraphique à 5 bits ; **ASCII** 7 bits [3], *American Standard for Information Interchange* ; EBCDIC...) étaient de ne représenter que le minimum de symboles nécessaires à l'anglais, à l'exclusion des diacritiques (accents, c cédille, etc) et sans tenir compte des autres graphies non latines. Dans certains cas, certains symboles pouvaient être obtenus par des séquence d'**échappement** [4].

1. *Interface Definition Language Compiler*, ou **IDL** compiler.
2. concept similaire aux fonctions de sérialisation et désérialisation des langages objets.
3. stocké sur un octet dont le **MSB** est 0
4. par exemple les entités HTML ou le LaTeX utilisé pour ce document

	2	3	4	5	6	7
0:		0	@	P	`	p
1:	!	1	A	Q	a	q
2:	"	2	B	R	b	r
3:	#	3	C	S	c	s
4:	$	4	D	T	d	t
5:	%	5	E	U	e	u
6:	&	6	F	V	f	v
7:	'	7	G	W	g	w
8:	(	8	H	X	h	x
9:	)	9	I	Y	i	y
A:	*	:	J	Z	j	z
B:	+	;	K	[	k	{
C:	,	<	L	\	l	|
D:	-	=	M	]	m	}
E:	.	>	N	^	n	~
F:	/	?	O	_	o	DEL

Code ASCII 7 bits, sans la plupart des caractères de contrôle (par exemple : `A` est `0x41`).

Code Latin-1 (ISO-88591), sans certains caractères de contrôle.

Sources : `man 7 ascii` et https://www.unicode.org/charts/PDF/U0080.pdf

FIGURE 6.2 – Codes ASCII et Latin-1 en hexadécimal

6.4.2 Jeux de caractères régionaux ou nationaux

L'approche insuffisante de l'ASCII a été ensuite complétée sur 8 bits par des symboles supplémentaires des langues européennes, de manière plus ou moins compatible aboutissant aux standards régionaux de la famille **ISO-8859** comme **ISO-8859-1**[5] pour les langues ouest-européennes (français, allemand, espagnol) dès la fin des années 1980.

Certains constructeurs ont adopté ces normes. D'autres ont préféré développer leurs propres jeux de caractères (p.ex. MS-DOS **CP437**, variant très fortement par rapport au standard, puis Microsoft **Windows-1252** – ce dernier jeu étant très proche de ISO-8859-15, à part la fameuse apostrophe incorrecte de Microsoft Word).

D'autres standards régionaux ont ensuite été adoptés pour les autres groupes linguistiques, y compris des encodages ne représentant que partiellement les langues à idéogrammes.

6.4.3 Unicode

6.4.3.1 Le répertoire universel

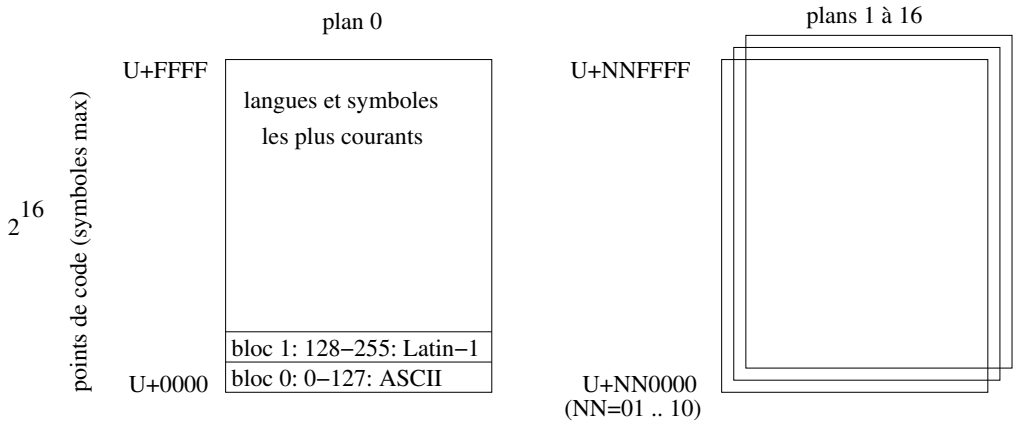
La solution proposée par les jeux de caractères régionaux n'était pas satisfaisante, pour les langues avec des milliers de caractères ou si l'on désirait utiliser dans le même document des symboles de jeux différents (p.ex. symboles de l'alphabet grec dans un texte mathématique en français) : un jeu *universel* de caractères devenait nécessaire, sur bien plus que 8 bits.

5. **ISO-8859-15** en est une extension contenant notamment le symbole monétaire de l'Euro.

Ce jeu, **JUC** (ISO/CEI-10646), a été complété par la spécification complète de la graphie (du dessin) des caractères et ainsi qu'une normalisation sémantique [6] dans la norme **Unicode**. Dans cette norme, chaque symbole différent de chaque langue possède un identifiant numérique unique hexadécimal, le **point de code**, préfixé par le préfixe `U+`. Chaque point de code se trouve au sein de blocs de codes situés dans des plans Unicode.

6.4.3.2 Découpage en plans et blocs

L'espace Unicode comporte un peu moins de 21 bits, ce qui correspond à un peu plus de 800'000 points de code utilisables. Il est découpé en 17 plans de $2^{16} = 65536$ points de codes, eux-mêmes découpés en blocs de 128 caractères :



Par exemple, le premier plan (plan 0, plan multilingue de base, points de code `U+0000` à `U+FFFF`) contient les caractères les plus utilisés (dont le code ASCII dans le premier bloc et Latin-1 (ISO-8859-1) dans le deuxième, puis de nombreuses langues occidentales, sémites, asiatiques, y compris les caractères chinois les plus courants et les emojis). Les plans suivants contiennent des symboles moins courants, les langues anciennes et symboles. Il existe même des zones à usage privé (non normé) dans le premier plan [7] et des plans entiers privés à disposition.

6.4.3.3 Encodages

L'espace Unicode est très grand : les premiers jeux de caractères étaient soit peu optimisés en longueur, soit ne permettaient pas de représenter tous les points de code (UCS-x, x désigne le nombre d'octets). Ensuite, ont été défini les jeux UTF-x (x désigne le nombre de bits de base), avec UTF-8, UTF-16 et UTF-32 qui sont les encodages utilisés aujourd'hui :

6. p.ex. interaction positionnelle entre symboles : un e aigu a son propre point de code, mais il existe aussi un point de code accent aigu que l'on peut combiner avec le point de code du e simple, voir la section 6.4.3.6 en page 102.

7. p.ex. APPLE définit `U+F8FF` comme une pomme croquée pour les polices Apple mais dans d'autres polices c'est le logo Windows ou le symbole Euro...

jeu	taille	description	remplace
UTF-8	1 (ASCII), 2, 3 ou 4 octets	très populaire, longueur variable optimisée, proposé par Ken THOMPSON – auquel on doit aussi Unix, la commande `grep`, les systèmes d'exploitation Plan 9 & Inferno et le langage Go	UCS-1
UTF-16	2 octets (16 bits, pour le premier plan 0) ou 4 octets (32 bits, pour les plans 1 à 16)	a remplacé UCS-2 qui ne pouvait représenter que le 1er plan et était utilisé p.ex. dans les SMS dès qu'il y avait un caractère non ASCII : la Chine, en obligeant les applications, sur son territoire, à supporter également le plan 1, a rendu UCS-2 obsolète	UCS-2
UTF-32	4 octets (32 bits)	représente tout l'espace Unicode mais de manière inefficace (fichiers 4 fois plus grands)	UCS-4

6.4.3.4 UTF-8

L'encodage[8] aujourd'hui le plus répandu est à taille variable minimale, l'**UTF-8**. Son principe est d'utiliser les 8 bits de chaque octet et d'être codé sur 1 octet pour les caractères du code ASCII, et sur 2 à 4 octets pour les caractères ne figurant pas dans le jeu ASCII, en fonction du nombre de bits significatifs selon le tableau ci-après.

représentation binaire UTF-8	bits significatifs x
0xxxxxxx	1 octet codant 1 à 7 bits (ASCII)
110xxxxx 10xxxxxx	2 octets codant 8 à 11 bits
1110xxxx 10xxxxxx 10xxxxxx	3 octets codant 12 à 16 bits
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4 octets codant 17 à 21 bits

Ce codage dynamique, basé sur les formats ci-dessus (composés d'un préfixe en fonction du nombre d'octets nécessaires pour les bits significatifs[9] à coder, puis d'octets de continuation débutant par 10) permet de déposer les bits du point de code, complétés à gauche par des zéros éventuels.

Il évite d'augmenter de manière significative la taille d'un fichier texte ne contenant pas de nombreux symboles non ASCII. Il est à codage minimal : il est *interdit* de coder plus long que nécessaire.

Les bits obligatoires de préfixe débutant chaque octet permettent de resynchroniser le flot de données, en cas de perte d'un octet par exemple.

Le tableau ci-après montre quelques exemples de symboles codés dans divers jeux usuels :

symbole	ASCII	ISO-8859-1	point de code Unicode	UTF-8 (hexadécimal)
A	65 (0x41)	65 (0x41)	U+0041	0x41
é	-	233 (0xE9)	U+00E9	0xC3 0xA9
α	-	-	U+03B1	0xCE 0xB1
中	-	-	U+4E2D	0xE4 0xB8 0xAD
⌟	-	-	U+1FB7F	0xF0 0x9F 0xAD 0xBF

8. on dit aussi code, codage ou jeu de caractères
9. par exemple 001000 a 4 bits significatifs

Pour le 3^{e} exemple, le point de code U+03B1 s'écrit en binaire 1110110001 (10 bits significatifs), la forme retenue est donc 110xxxxx 10xxxxxx et sa valeur est donc 11001110 10110001 ce qui est bien 0xCE 0xB1.

Pour l'avant-dernier exemple de l'idéogramme chinois signifiant *milieu*, représenté par un rectangle transpercé d'un trait de bas en haut, la valeur binaire de la séquence UTF-8 vaut 11100100 10111000 10101101, les bits à représenter sont donc 0100111000101101, le point de code Unicode est donc bien U+4E2D.

Enfin, le dernier est un symbole historique lié au Commodore 64 [10].

6.4.3.5 Endianness

Comme les encodages UTF-16 et UTF-32 ont des variantes big/little endian [11], un point de code débutera les fichiers correspondants (`U+FEFF` – **BOM**, *Byte Order Mark*) pour déterminer l'endianness.

Un fichier codé UTF-8, même s'il n'a pas de notion de big ou little endian, peut optionnellement aussi commencer par ce point de code, encodé en UTF-8 `0xEF 0xBB 0xBF`, ce qui est en fait inutile et pose des problèmes à certains logiciels.

6.4.3.6 Normalisations NFC et NFD

Unicode prévoit des symboles de combinaison pour créer des symboles composites : par exemple on peut écrire un `e aigu` soit comme un seul point de code Unicode `U+00E9`, ou comme deux points de code : `U+0065` (`e` en ASCII) et `U+0301` (caractère de composition accent aigu).

Cela crée bien sûr une ambiguité (rien que pour comparer des chaînes). Pour cette raison, deux *normalisations* ont été définies :

NFC la forme compacte : un seul point de code Unicode

NFD la forme décomposée : plusieurs points de code Unicode

Tout logiciel devrait accepter NFC ou NFD en entrée, puis transformer en une seule des deux normalisations, avant d'effectuer des opérations de validation ou de comparaison, par exemple.

NFC est la forme la plus courante ; toutefois, un avantage de la forme NFD est la facilité pour supprimer les diacritiques en filtrant les points de code supérieurs à U+007F.

6.4.4 Transparence du transport de données

Jusqu'ici, il a été supposé que la transmission de tous les 8 bits de données étaient possibles, et donc que, quelque soit le jeu de caractère utilisé, les données arriveraient sans encombre au destinataire.

En pratique, cela n'est pas toujours le cas [12] : une idée qui n'est pas devenu un standard était l'**UTF-7** [13] : aujourd'hui on utilisera plutôt par exemple des encodages de transfert MIME comme **quoted-printable** ou **base64** (voir section 7.2.5.1.2 en page 122), ou des échappements.

10. https://en.wikipedia.org/wiki/Symbols_for_Legacy_Computing, U+1FB00 – U+1FBFF
11. qui diffèrent par une inversion des octets dans chaque mot de 16 bits
12. certains protocoles de couche 7 comme SMTP ou HTTP ne le supportent pas par défaut, par exemple
13. RFC-2152, basé sur l'**échappement** des caractères non ASCII

6.4.5 Déterminaison du jeu utilisé

Le standard **MIME** (voir 7.2.5.1.2 en page 121) permet de typer le jeu de caractères choisis (**charset**) dans une partie d'un courrier électronique ou dans un document du Web transmis par HTTP via l'entête **Content-Type** ou spécifié dans un entête `meta`.

Lorsque l'on ne spécifie pas de jeu de caractère, ASCII 7 bit voire p.ex. pour l'HTML ISO-8859-1 sont supposés. Certains langages ou outils, notamment du monde XML, exigent, ou comme le HTML5, recommandent, l'UTF-8 comme jeu standard.

6.5 Compression

L'objectif principal[14] de la compression est de réduire la taille des données à transmettre, dans un compromis entre débit disponible et puissance CPU, voire mémoire vive (RAM) à disposition.

La compression peut-être sans perte (données informatiques, données multimédia à retraiter / éditer), ou avec perte (principalement pour le transfert et la restitution humaine, car les sens humains sont imparfaits).

La compression sans perte réduit la structure des données en s'attaquant soit à la répartition individuelle des symboles (inégalité de fréquences d'apparition : on abandonne l'idée d'un code de taille fixe et donc on code sur moins de bits les symboles les plus fréquents, pour qu'en moyenne la taille des données diminue) et/ou aux répétitions fréquentes dans les données (codage par sous-chaînes / dictionnaires, encodage des répétitions, modélisation des probabilités d'apparition d'un symbole après un autre / probabilités conditionnelles).

La compression avec perte supprime les informations invisibles à l'humain (résolution de couleur, résolution spatiale, détail du mouvement, fréquence audio maximum, etc). Les deux techniques (avec ou sans perte) sont souvent combinées pour les données multimédia à restitution humaine.

Ces éléments seront traités plus en détail dans le cours ISC2 IL/ID Protocoles et réseaux.

14. un objectif secondaire : si l'on compresse avant de chiffrer, la résistance aux attaques sur le chiffrement de ces données sera meilleure, en raison de la la destructuration – suppression de la redondance – induite par la compression ; compresser *après* chiffrement n'a pas de sens, sauf si l'on veut tester la qualité du chiffrement : des données mal chiffrées se compresseront mieux que si elles sont bien chiffrées.

6.6 Chiffrement et signature numérique

Ces concepts font plutôt l'objet du cours Cryptographie IL/ID de 3^e année : cependant, il est évident qu'ils sont aussi utilisés journellement sur Internet et qu'une compréhension minimale bénéficie à tous (y compris IE) dès la 1ère année, en complément au cours Sécurité pratique.

6.6.1 Principes de base

6.6.1.1 Chiffrement et signature numérique

Tout chiffrement est toujours basé sur un **algorithme** et une (ou plusieurs) **clés**.

Originellement, toute la difficulté d'un algorithme de chiffrement tenait dans son secret : par exemple le code de César [15], une fois que l'on en a compris le principe, se casse pour toute clé existante par une analyse statistique (certaines lettres étant plus fréquentes que d'autres en français, p.ex.). Aujourd'hui, on considère généralement que les algorithmes devraient être **connus et largement publiés** (pour pouvoir déterminer leur résistance, leurs biais ou trouver des erreurs d'implémentation [16]), et que c'est dans la génération et la gestion des clés que réside toute la sécurité.

Le chiffrement est basé sur des résultats mathématiques qui permettent de trouver des fonctions *trappes* (difficilement inversibles). Toute la difficulté est de trouver un algorithme qui produise un chiffrement aisément (à l'aide d'une clé), mais ne puisse pas être inversé dans un temps raisonnable par un attaquant sans celle-ci, même en connaissant l'algorithme.

Cette notion de *difficulté* est toute relative : la taille de clés nécessaire [17] à une bonne sécurité augmente régulièrement avec la puissance des ordinateurs et pourraient éventuellement devoir brusquement changer avec un bouleversement d'architecture (p.ex. l'ordinateur quantique) – ce qui pourrait éventuellement nécessiter des nouveaux paradigmes [18], comme par exemple le chiffrement quantique et du matériel spécifique [19].

La **signature numérique** [20], qui a pour but d'authentifier des transactions et est basée sur la propriété de symétrie du chiffrement asymétrique [21], est aujourd'hui considérée, si implémentée de manière correcte à tous les niveaux (algorithmes, code, procédures, ...) d'une valeur similaire, voire supérieure, à la signature papier.

6.6.1.2 Fonction de hachage

Le but d'une fonction de hachage est de fournir un *résumé* – une empreinte – d'un message, sur beaucoup moins de bits.

---

15. un simple décalage de N position dans l'alphabet, N étant la clé
16. si aucun bug n'a jamais été trouvé dans un logiciel, cela veut dire que l'on n'a pas bien cherché
17. on considère aujourd'hui que 2048 bits en chiffrement asymétrique et 128 bits en chiffrement symétrique offrent une bonne sécurité avec certains algorithmes bloc – ajouter 1 bit signifie, en symétrique, doubler la complexité
18. https://en.wikipedia.org/wiki/Post-quantum_cryptography
19. ce qui pourrait inverser la tendance de tout est logiciel de ces dernières années et mener à une *decommoditisation* du matériel informatique
20. ne pas confondre avec la **signature électronique** qui, si à l'origine avait une signification identique, désigne aujourd'hui, selon la législation européenne, un simple contenu numérique (image, son, etc) confirmant par exemple réception d'un document
21. une clé chiffre, l'autre déchiffre ; et vice-versa : dans un cas il s'agit de chiffrement, et dans l'autre on l'utilise plutôt dans la signature numérique

Plus formellement, les fonctions de hachage sont des fonctions mathématiques que l'on peut assimiler à des projections (surjections). Cela signifie qu'à partir d'un ensemble de départ (formé de tous les messages de N bits ou moins possibles, on fait correspondre un ensemble d'arrivée de hachages de M bits (avec en général $M << N$). Chaque message doit donc correspondre à un hachage (anglais : *hash*, empreinte). Par contre, et par propriété de la projection, un hachage correspond forcément à plusieurs messages.

On distinguera entre les fonctions de hachage classiques (qui servent p.ex. à optimiser la gestion d'indexes en dispersant de manière appropriée les valeurs sur un tableau) et les fonctions de **hachage cryptographique** qui ont des propriétés de résistance aux attaques cryptographiques :

— grande difficulté de retrouver le message m à partir d'un hachage $H(m)$ donné : résistance à l'**attaque préimage**
— de même, difficulté de retrouver un message différent m_2 donnant le même hachage $H(m) = H(m_2)$ (seconde préimage)
— enfin, résistance aux **collisions cryptographiques** : difficulté de trouver deux messages différents donnant un hachage identique

Les hachages cryptographiques sont donc des outils permettant d'assurer qu'un message n'a pas été altéré, sans devoir transmettre ce message en clair.

Exemples de fonctions de hachage cryptographiques : MD5 (compromis, notamment pour les collisions sur des données maîtrisables), SHA-256 (famille SHA-2)...

6.6.1.3 Notion de confiance

La signature numérique et plus globalement le chiffrement en général font appel à des notions de confiance : en effet, comment assurer qu'un message vient bien de son expéditeur (**non répudiation**) et n'a pas été altéré (**intégrité**). Dans certains cas, l'**horodatage** (estampillage) ou la présence d'un tiers certifiant est nécessaire pour exécuter des transactions en toute confiance.

La confiance en un partenaire est définie par deux niveaux : la confiance dans les processus internes de sécurité du partenaire et l'assurance qu'il est bien celui qu'il prétend être.

Avec la signature numérique, on vérifie qu'un message donné – ou son hachage – provient bien de l'expéditeur supposé, en vérifiant de proche en proche que la clé utilisée pour signer est bien associée à l'identité de l'expéditeur, ceci par un réseau de confiance ou un chemin de confiance hiérarchique.

6.6.1.4 Aléatoire

Les cryptosystèmes font intervenir le hasard à plusieurs niveaux : pour la création de clés temporaires, pour la génération de clés de chiffrement à plus long terme, pour des paramètres de protocoles d'échanges de clé, etc.

Les générateurs[22] aléatoires, s'ils sont utilisés pour des applications cryptographiques, doivent remplir des critères bien plus sévères que ceux utilisés[23] pour des programmes informatiques simples (jeux, voire simulations statistiques).

Une des plus grandes vulnérabilités d'un système cryptographique est souvent lié au générateur aléatoire. S'il n'est pas bon, les clés générées ne seront pas adéquates. On utilise souvent des

22. n'est-ce pas étrange de prétendre pouvoir générer des valeurs aléatoires ? les meilleurs sont basés sur des sources physiques ou des notions de pool d'entropie, p.ex. le **/dev/random** de Linux
23. PRNG – Pseudo Random Number Generators

sources physiques et événementielles pour améliorer la qualité d'un générateur.

6.6.1.5 Lien à la compression

La compression de données agit sur la redondance des données (p.ex. les répétitions, ou la statistique du langage considéré). Ce faisant, elle réduit en général la taille des données et produit une sortie qui est moins susceptible d'être attaquée par cryptoanalyse statistique. C'est pourquoi la compression précède souvent le chiffrement. Un bon algorithme de chiffrement devrait d'ailleurs produire une sortie difficile à compresser.

6.6.1.6 Modèle en couche

En pratique, du point de vue du développeur, le support du chiffrement et de la signature numérique est implémenté comme une couche (anglais : *layer*) au-dessus de TCP : citons par exemple les implémentations **SSL** (*Secure Socket Layer*) et **TLS** (*Transport Layer Security*).

La plupart des applications nécessitant de la cryptographie utilisent donc ces bibliothèques directement (API C par exemple), et ensuite bénéficient d'un canal sécurisé implémenté au-dessus de TCP.

6.6.2 Types de chiffrement

6.6.2.1 Chiffrement symétrique ou à clé secrète

On parle de **chiffrement symétrique** lorsque la clé de chiffrement est la même que la clé de déchiffrement (secret partagé, clé secrète).

Le chiffrement symétrique est en général rapide, simple, et ne nécessite pas de clés très longues. On en trouve deux types principaux : ceux travaillant octet par octet (stream) ou par blocs.

De nombreux algorithmes existent comme par exemple AES, 3DES, IDEA, etc, utilisables dans ces deux modes (stream ou bloc). Ces algorithmes, dans la mesure où ils sont codés et déployés de manière sûre, sont sujet, en plus, à des attaques cryptographiques potentielles, p.ex. : **brute force**, **known plaintext**, **known ciphertext**... En effet, certains algorithmes symétriques de type stream, ou les algorithmes de type bloc qui ne perturbent pas la clé avec un vecteur d'initialisation, sont vulnérables à des attaques statistiques directes, voire même basées sur un texte en clair partiellement connu [24].

Le problème principal du chiffrement symétrique est la distribution des clés, qui doit se faire par un canal sûr (chiffré et signé), par exemple une clé USB remise en mains propres. Un échange **Diffie-Hellman**, qui permet de créer une clé de session symétrique commune aux deux partenaires sans transfert de celle-ci, est également possible, mais il ne protège pas contre une attaque par relais (**MitM**) où deux demi-tunnels Diffie-Hellman sont créés : il y manque l'authentification du partenaire.

L'avantage principal du chiffrement symétrique, outre sa simplicité et sa robustesse même pour des tailles de clés raisonnables, est qu'il est souvent possible de chiffrer en matériel.

24. que se passe-t-il si vous faites un XOR entre la clé et une suite de NULs (ASCII 0) ?

6.6.2.2 Chiffrement asymétrique ou à clé révélée ou publique

On parle de **chiffrement asymétrique** lorsque la clé de chiffrement est différente de la clé de déchiffrement. L'algorithme est donc dit asymétrique. Les deux clés sont générés ensemble et sont liées par des propriétés inhérentes à l'algorithme considéré. Une clé est appelée privée ou secrète (K_s), et l'autre publique (K_p). La clé privée doit être précieusement conservée cachée, et la clé publique peut être diffusée globalement, éventuellement via un **certificat** (voir section 6.6.5 en page 108). Les algorithmes à clé publique sont par exemple RSA ou ElGamal.

Par une propriété de symétrie, on peut chiffrer avec une des clés et déchiffrer avec l'autre, et vice-versa. Par exemple si Alice veut envoyer un message m à Bob, elle utilisera la clé publique de Bob : $K_{B_p}(m)$ puis Bob déchiffrera avec sa clé secrète pour retrouver le message m : $K_{B_s}(K_{B_p}(m)) = m$. On verra que chiffrer avec sa clé publique est utile pour la signature numérique (voir section 6.6.3 en page 107) et plus généralement pour authentifier un message.

En pratique, on préfère chiffrer les données avec une clé symétrique : en effet c'est plus rapide que le chiffrement asymétrique, plus sûr (en particulier avec des variations de clés) et c'est implémentable en matériel. Toutefois, le chiffrement symétrique seul se heurte à la complexité de l'échange des clés sécurisé. On combinera alors avec le chiffrement asymétrique pour assurer que le partenaire est le bon afin d'éviter l'attaque du relais (**MitM**, *Man in the Middle*). La clé symétrique sera renouvelée régulièrement par le protocole.

Il y a plusieurs techniques pour créer la clé symétrique : une première méthode est de générer aléatoirement [25] une clé de session *symétrique*, qui est chiffrée en asymétrique par la clé publique du destinataire [26]. L'inconvénient est que si une clé privée est volée par la suite [27] ou si l'algorithme asymétrique devient cassable par l'augmentation de la puissance des machines, un attaquant ayant stocké tout le trafic du passé peut le déchiffrer par la suite.

Pour éviter ces deux vulnérabilités futures, une variante plus moderne, appelée **PFS** (*Perfect Forward Secrecy*), consiste à créer un secret partagé entre les deux systèmes communiquants par le protocole **Diffie-Hellman**, en y ajoutant une authentification du partenaire avec du chiffrement asymétrique afin d'éviter l'attaque du relais (**MitM**) vue précédemment. La clé de session (le secret partagé) n'est jamais transmis sur le réseau. Un attaquant potentiel ne pourra donc pas obtenir la clé de session (et donc ne pourra déchiffrer le trafic), même en ayant stocké tout le trafic et même en réussissant, plusieurs années après, une attaque contre la clé asymétrique (vol ou cassage). PFS assure donc une confidentialité, même en cas de développements techniques futurs ou de vol d'une clé privée (mais pas l'authenticité dès le moment que la clé privée est compromise) !

6.6.3 Signature numérique

Le but de la signature numérique est de pouvoir s'assurer que le partenaire est le bon (**authenticité**) et que le message n'a pas été modifié par un tiers (**intégrité**). En effet, dans l'exemple ci-dessus, rien n'empêche Charles de faire croire à Alice que la clé publique de Bob est K_{C_p} et donc, Charles pourra, par une attaque du relais (**MitM** attack), déchiffrer, prendre connaissance du message, le modifier, et le transmettre chiffré normalement à Bob.

Pour assurer que le message n'a pas été modifié, le plus simple n'est-il pas de le hacher : $H(m)$ [28]. Et pour éviter que Charles puisse modifier le message et l'empreinte, il faut faire

25. avec un bon générateur aléatoire, p.ex. basé sur un pool d'entropie
26. ou les clés publiques des différents destinataires, chacune chiffrant sa copie de la clé symétrique
27. p.ex. Heartblead, 2014, voir http://heartbleed.com/
28. empreinte de m par la fonction de hachage H

signer l'empreinte par Alice de manière à ce que Bob puisse la vérifier, mais que personne ne puisse la modifier. Mais comment faire ?

La propriété de symétrie du chiffrement asymétrique est le fondement de la signature numérique : en effet, si Alice chiffre $H(m)$ avec sa clé **privée**, Bob peut déchiffrer l'empreinte avec la clé publique d'Alice : Charles, ne connaissant pas la clé privée d'Alice ne pourra alors pas produire une empreinte chiffrée compatible avec ses changements !

Alice enverra donc $(K_{B_p}(m), K_{A_s}(H(m)))$ et Bob calculera $(K_{B_s}(K_{B_p}(m)), K_{A_p}(K_{A_s}(H(m))))$ ce qui vaut (m, H(m)) et il pourra donc lire le message et vérifier son empreinte.

Evidemment, on a un problème : autant la clé publique de Bob aurait pu être falsifiée par Charles, autant celle d'Alice peut l'être ! Le problème peut être résolu si Bob et Alice se transmettent leurs clés publiques par un canal sûr, mais cela est très compliqué, en particulier sur un grand réseau où les partenaires ne se connaissent pas personnellement.

La solution cryptographique de diffusion des clés publiques de manière sûre (**PKI**, **Public Key Infrastructure**) est basée sur les certificats.

6.6.4 Message Integrity Code (MIC)

Si la signature numérique telle que vue ci-dessus permet d'assurer l'authenticité et l'intégrité d'un message, elle peut être complexe à mettre en oeuvre, trop coûteuse ou risquée (en particulier si l'attaquant peut choisir les messages à chiffrer).

On peut donc généraliser le problème que la signature résout (authenticité et intégrité) et aboutir au concept de MIC [29] (*Message Integrity Code*, code d'intégrité de message).

Les MICs doivent résister aux attaques de message en clair choisi par l'attaquant (**chosen-plaintext**). Ils peuvent être basés sur des systèmes de chiffrement symétriques seuls ou sur une combinaison avec des hachages cryptographiques : dans ce dernier cas on parle de **HMAC** (hachage cryptographique + chiffrement à clé secrète de session). Les clés symétriques de session peuvent être authentifiées régulièrement par PKI (voir le processus en section 6.6.2.2 en page 107).

6.6.5 Certificats

6.6.5.1 Principes

Un certificat associe une *identité* d'un domaine quelconque (p.ex. le monde réel) à une clé publique, ce qui permet ensuite de vérifier l'authenticité de la clé publique associée, et donc par exemple d'un message signé numériquement par la clé privée correspondante.

Un certificat contient au moins ces informations :

- une identité **DN** (*Distinguished Name*), composée de plusieurs attributs externe, dont par exemple le **CN** (*Canonical Name*), www.alphanet.ch, schaefer@alphanet.ch, ou Marc Schaefer, passeport numéro XXX)
- la clé publique de cette identité
- des dates de validité
- l'emplacement éventuel d'un système dynamique de révocation (CRL)
- la signature numérique de ce qui précède par une autorité de certification (par sa clé privée, forcément), identifiée par son DN

29. on peut aussi parler de *Message Authentification Code*, ou MAC – mais ne pas confondre avec l'adresse MAC de la sous-couche Media Access Control

Il faut cependant faire confiance à l'autorité de certification qui a signé le certificat ! Ce qui peut s'assimiler au problème de la poule et de l'oeuf, si l'on n'a pas à sa disposition un mécanisme de confiance approprié. Certaines autorités ne certifient qu'une partie du certificat (p.ex. uniquement le CN).

6.6.5.2 Confiance

6.6.5.2.1 Réseau de confiance On parle de réseau de confiance lorsque les certificats sont générés par qui le veut bien et publiés d'une manière ou d'une autre (p.ex. serveur de clé ou *keyserver*).

C'est le principe du logiciel GPG (qui suit la norme OpenPGP), fréquemment utilisé pour la signature numérique d'e-mail ou de paquets logiciels.

Les procédures permettant à une personne A de signer la clé publique d'une personne B sont bien documentés et devraient éviter des erreurs ou compromissions : si elles surviennent, il est possible de générer une annonce de révocation (publiée sur les serveurs de clé), ou, plus simplement, de se baser sur le nombre de signatures provenant de sources différentes auxquelles on fait confiance pour établir un degré de confiance.

Cette propriété unique de validation en réseau (communautaire, chacun jouant le rôle d'une autorité de certification) est ce qui fait un réseau de confiance. La valeur ajoutée par rapport aux simples chemins de confiance est que la compromission d'un signataire ne compromet pas immédiatement l'ensemble des clés qu'il a signé.

Toutefois, la publication sur les serveurs de clés a mené à des attaques de déni de service (**DoS**) ce qui a montré les limites d'un réseau de confiance ouvert.

X.509 n'utilise pas les réseaux de confiance, mais les chemins de confiance (voir ci-après).

6.6.5.2.2 Chemin ou hiérarchie de confiance Dans le modèle classique des certificats SSL/TLS du Web, et plus généralement pour les certificats **X.509**, tout le système repose sur une (ou plusieurs) autorité de certification qui, présente dans les navigateurs sous forme d'un (ou plusieurs) certificat(s) racine(s) (**root certificate**), permet de vérifier les certificats présentés.

Cette certification peut se faire de manière hiérarchique : il suffit qu'une chaîne de confiance mène du certificat présenté à un des certificats racines. En pratique, il n'y a que les certificats intermédiaires qui sont signés par le certificat racine, pour des raisons de sécurité.

Par le principe même d'une chaîne de confiance, dès qu'un certificat racine est compromis (ou n'importe quel certificat intermédiaire signé par un des certificats racines), le système s'effondre.

Pour réduire ce problème d'effondrement de la confiance, voici ce que X.509 propose aujourd'hui :

- détecter des abus de la part des autorités de certification peut aboutir à la suppression des certificats racines (des navigateurs et systèmes d'exploitation – cela arrive régulièrement)
- le protocole OCSP (RFC-6960) permet la validation en-ligne de certificats, ainsi que la gestion de listes blanches de certificats intermédiaires
- il est possible d'indiquer dans le DNS (**CAA**) quelles autorités de certifications (CA) peuvent signer les certificats d'un domaine donné – évidemment plus sûr en combinaison avec **DNSSEC**

Toutefois, X.509 continuer à se baser sur un chemin de confiance (la recherche académique continue sur l'utilisation de réseaux de confiance).

6.6.5.3 Autorité de certification (CA)

L'autorité de certification est responsable d'assurer que ses clés ne sont pas compromises, de renouveler régulièrement ses clés et certificats, et globalement d'être gérée de manière compatible avec la sécurité. Un minimum est en général exigé par les éditeurs de clients WWW pour installer le certificat racine de l'autorité dans le client, et surtout une contribution financière de plusieurs centaines de milliers de francs.

Toutes les autorités de certifications ne sont pas comparables, et elles ne délivrent pas les mêmes types de certificats (ce qui se ressent par exemple par des couleurs différentes dans les clients WWW – et des prix différents). En effet, si certaines autorités vérifient les passeports, les registres du commerce, etc, d'autres ne vérifient que la capacité du demandeur à répondre aux e-mails du domaine considéré.

Parfois, les autorités sont compromises (piratage, erreur de manipulation) et cela doit être géré par les mises à jour du client WWW, ou, s'il s'agit d'un certificat isolé, des listes de révocation fournies par les CA.

6.6.6 Blockchain

6.6.6.1 La confiance sans tiers garant : le consensus décentralisé

Dans les systèmes distribués où les divers partenaires ne se font pas confiance, le premier réflexe est de désigner un tiers-garant, qui, tel un notaire virtuel, certifiera les transactions. Par exemple, une autorité de certification (voir section 6.6.5.3 en page 110) est un tiers de confiance qui garantit qu'une certaine clé publique correspond à un certain nom de domaine, voire même à une identité du monde réel. De plus, il y a souvent nécessité de stocker des informations dans un journal horodaté et accessible de tous, à fin de vérification : tous ces éléments peuvent être résumés par le concept de **consensus** distribué.

Comment faire pour assurer un consensus sans tiers-garant ni point central ? C'est cette problèmatique qui est résolue par la **blockchain**.

Idéalement, une information fait partie du consensus lorsque plus de 50% des partenaires [30] sont d'accord avec ce fait. Cette information peut être des transactions de monnaie virtuelle ou des **smart contracts** [31].

6.6.6.2 Principes de base

Le principe de cette technique est de remplacer la signature numérique (qui nécessite une chaîne de confiance ou un réseau de confiance, voir section 6.6.5.2 en page 109) par une preuve de travail (**proof of work**).

Lorsqu'un partenaire veut intégrer une nouvelle information au consensus, il la propose au réseau, qui l'intègre à un bloc contenant d'autres informations, dont le hachage d'un bloc ayant précédemment [32] été miné. Ensuite, des **mineurs** vont tenter de résoudre un problème coûteux en calcul (CPU) le plus rapidement possible.

30. qui doivent stocker l'ensemble des transactions, l'ensemble du consensus
31. du code exécuté lorsque une condition est atteinte
32. qui lui même contient le hachage d'un autre bloc, et récursivement jusqu'au bloc de base, appelé le **genesis block**, qui fait partie du logiciel

Le premier mineur qui arrive à résoudre le problème [33] gagne [34] et reçoit une contre-partie, par exemple une valeur en en monnaie virtuelle (p.ex. **Bitcoin**) – qui est la motivation de miner en premier lieu.

A tout moment, le réseau contient donc plusieurs chaînes de bloc minées, potentiellement incompatibles, et plus la chaîne est longue, plus il y a de probabilité que cette chaîne fasse partie in fine du consensus. Il peut y avoir plusieurs chaînes de bloc qui temporairement sont incompatibles, et parfois des annulations : plus la chaîne est longue moins le risque existe.

Pour compenser l'augmentation de puissance des machines ou l'intérêt d'utiliser des systèmes de plus en plus dédiés (fermes, GPU, FPGA, ASIC), la difficulté du problème augmente [35] avec le nombre de blocs minés.

En pratique, vu la taille du consensus, et l'augmentation des ressources avec le temps, les clients du protocole n'implémentent en général pas le service ou le minage. Ils passent souvent par des proxies (qui peuvent être source de vulnérabilités).

6.6.6.3 Problèmes

Les problèmes fondamentaux de cette technologie sont les suivants :

- gaspillage de ressources en raison du **proof of work** [36]
- pas de **scalability** : le consensus (journal des transactions) est de plus en plus grand
- plus les mineurs s'organisent en équipe contrôlées par un répartiteur, plus le risque que plus de 50% de la puissance de minage soit contrôlé dans les mêmes mains, et alors des attaques sur le consensus sont possibles
- un *fork* du logiciel est généralement un fork du consensus
- technologie complexe, avec des vulnérabilités potentielles dans toute la chaîne (algorithmes, implémentations, échanges...)
- chaque fork du consensus crée, pendant un certain temps, un jeu de l'avion, en particulier si le bloc de base (**genesis block**) est dans des mains actives : de nombreuses monnaies alternatives existent, jouets de spéculation financière
- avec les monnaies virtuelles, ce qui est stocké dans le consensus ce sont des transactions de transfert entre clés publiques : il faut donc protéger sa clé privée du vol
- le journal des transactions est par définition public : le système peut offrir au mieux du pseudonymat, mais pas de l'anonymat

A ces problèmes de base, s'ajoutent des problèmes pratiques : vu la charge de gestion du consensus [37], il faut une incitation de plus en plus grande, donc des frais de transaction, pour que les mineurs travaillent en priorité sur nos demandes. De plus, les clients passent souvent par des services intermédiaires (exchanges) qui peuvent être piratés. Enfin, bien souvent pour les transactions commerciales simples, le temps d'attente pour que le consensus soit complètement garanti est remplacé par un certain nombre de confirmations : plus les mineurs seront contrôlés, plus le risque d'un **double spending** existe.

33. par exemple ajouter de l'aléatoire au bloc jusqu'à ce qu'un hachage cryptographique donne un résultat d'une certaine forme
34. les autres mineurs en concurrence ont perdu leur temps – d'où l'intérêt de joindre des équipes de mineurs
35. par exemple, il faut trouver un hachage avec de plus en plus de zéros au début
36. le minage de Bitcoin est évalué en 2023 à environ 122 TWh alors que l'extraction et le raffinage de l'or consommerait le double, voir : https://www.rockitcoin.com/blog/bitcoin-energy-consumption-vs-gold-and-banks/ qui mentionne aussi l'impact de l'obsolescence potentielle d'une partie de l'industrie bancaire, forte consommatrice d'énergie ; aussi, certains prétendent que le Bitcoin pourrait rentabiliser les pics de production d'énergie renouvelable – sujet complexe s'il en est !
37. et la création de monnaie à chaque bloc miné de plus en plus faible, dans le Bitcoin

6.6.6.4 Solutions et avenir

La technologie en est encore à sa phase de recherche : de nombreux projets existent, comme par exemple Ethereum, qui a proposé, de modifier le concept de **proof of work** par l'implication dans l'écosystème des partenaires (**proof of stake**), ou l'approche *Scalable Byzantine Reliable Broadcast*[38] de l'EPFL qui propose de changer le paradigme *tout le monde est suspect* vers l'approche plus économique *les fraudes sont détectées et communiquées à l'ensemble du réseau*.

Pour réduire la taille du journal, des points de synchronisation peuvent être utilisés, ou des proxies pour certifier beaucoup plus rapidement des transactions. Le **lightning network** est une surcouche qui permet plus de transactions exécutées plus rapidement.

Si certains voient dans le Bitcoin et la blockchain une technologie *disruptive* mettant en question la chaîne de valeurs, en particulier de tous les emplois de type *intermédiaire*, d'autres y voient une technologie intéressante, forte utile pour certaines applications, mais très perfectible, tout en ne présentant aucun intérêt pour beaucoup de domaines où la signature numérique simple ou multiple suffit.

38. https://web.archive.org/web/20210303173913/https://crypto.unibe.ch/talks/20170622-blockchain-ice.pdf et https://arxiv.org/pdf/1908.01738.pdf

Chapitre 7

Couche application (7)

7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison
1	Physique

Sommaire

Le but de ce chapitre est de présenter le rôle de la couche application (en anglais *application layer*), la couche la plus haute du modèle OSI, de manière fonctionnelle puis sur des exemples pratiques.

7.1 Architecture applicative Internet

7.1.1 Client/serveur ou peer-to-peer

Les protocoles applicatifs classiques d'Internet sont basés sur une **architecture client/serveur** [1]. Ils utilisent soit TCP soit UDP dans la **couche transport**. Le service doit être activé du côté serveur. Si c'est le cas un **daemon** [2] est actif sur le port standard du service concerné (**well-known port**, voir section 4.2.3 en page 86). Un daemon est un programme qui attend les demandes de connexion des clients et qui instancie le service demandé pour chaque nouveau client.

7.1.2 Communication par sockets TCP ou UDP

Le client TCP se choisit [3] un numéro de port libre chez lui et qui est en dehors de la plage réservée (donc > 1023). Il établit ensuite une connexion TCP à l'aide de l'adresse IP et du numéro de port standard du serveur.

Du côté serveur, la connexion est identifiée de façon unique par la combinaison adresse IP du client, port du client, adresse IP du serveur et port du serveur. Il faut les quatre éléments car un même ordinateur client peut utiliser plusieurs fois le même service (par exemple FTP) : sur le même serveur ou client, chaque connexion (**nexus**) doit être identifiable individuellement.

7.1.3 Format des échanges

Contrairement aux protocoles des couches supérieures OSI qui définissent des messages structurés en champs (p.ex. encodés avec **BER** (*Basic Encoding Rules*) en fonction d'une **syntaxe abstraite** comme p.ex. **ASN.1** – voir section 6.2.2 en page 96), les protocoles de la couche supérieure d'Internet utilisent souvent des commandes en format ASCII (texte). On parle alors de protocole de type **ASCII NVT** (*Network Virtual Terminal*, terminal virtuel réseau), ou de **mode d'échange ligne-à-ligne**. Il se peut même parfois qu'il faille faire attention à la transparence des données (voir section 6.4.4 en page 102).

Ceci permet de les simuler interactivement à partir d'un programme simple capable de d'établir une session TCP, de lire des caractères du clavier et de les envoyer sur la connexion TCP. La commande `telnet` (ou `nc`, network `cat` [4]) peut être utilisée [5] à cette fin.

Certains protocoles récents utilisent une structuration de données texte en format XML (p.ex. le protocole de messagerie XMPP). Certains protocoles utilisent encore des encodages classiques binaires ASN.1/BER (p.ex. SSL/TLS, SNMP) ou suivent des encodages binaires particuliers et donc ne sont pas des formats textes.

Lors de la conception d'un nouveau protocole, le choix du format des échanges est essentiel :

1. voir section 0.2.2.2 en page 7 ; certains protocoles plus modernes offrent des échanges distribués ou **P2P**, en se basant toutefois toujours sur les primitives client/serveur TCP et UDP pour les échanges proprement dits ; certains protocoles multimédia font usage de **multicast** UDP ; au sein des centres de données, des protocoles spécifiques comme **Datacenter TCP** sont de plus en plus déployés.
2. Disk And Execution Monitor – jeu de mot sur le mot anglais demon (démon)
3. en fait c'est le kernel qui, en absence d'appel à `bind(2)` en choisit un aléatoirement ce qui renforce la sécurité.
4. la commande UNIX cat(1) permet d'afficher (*concatenate*) un ou plusieurs fichier sur l'écran ou sur une redirection : netcat en est la variante réseau.
5. ou abusée, car le protocole TELNET n'est alors pas utilisé, seul un échange de données ligne-à-ligne NVT

1. la plupart du temps et en particulier quand les données sont utilisés sur le web, un format ASCII de type JSON sera recommandé
2. dans le cas où une intégration avec des outils entreprise est nécessaire, le format ASCII XML sera favorisé
3. si une interopérabilité avec des formats existants télécom est nécessaire, y songer (ASN.1/BER p.ex.)
4. si de la haute performance et du multiplateforme sont nécessaires, **Protocol Buffers**[6] pourrait être un bon choix moderne
5. dans les autres cas, il faudra réfléchir sur des critères comme le volume de données, la puissance de calcul, etc.

Dans le monde embarqué, on a plus tendance encore aujourd'hui à utiliser des formats de données binaires spécifiques : par exemple si des données sont transférées sur **LoRaWAN**, la longueur des paquets est très importante et le total journalier limité.

Un chiffrement est fortement recommandé dans le transport des données (p.ex. HTTPS).

7.1.4 Protocoles de type interrogatif

Bien souvent, les protocoles Internet de couches supérieures sont de type interrogatif, avec une commande envoyée par le client, suivie d'une réponse par le serveur : il est très facile de modéliser un protocole de ce type avec un automate, avec divers degrés de formalisme.

Cette réponse est souvent préfixée par un code de résultat (protocoles FTP, SMTP, IMAP, POP, HTTP...) à 3 chiffres et dont le premier indique le type du résultat (voir tableau ci-dessous, valable en particulier pour le SMTP et le HTTP). Parfois, des échanges de données typées ou non sont effectués sur la même connexion (SMTP, HTTP), ou sur une connexion supplémentaire initiée soit par le serveur (FTP mode actif) ou par le client (FTP mode passif).

préfixe du code d'erreur	**signification générale**
1xx	information
2xx	tout va bien
3xx	quelque chose manque (redirection, authentification, données...)
4xx	erreur côté client (HTTP), ou erreur temporaire (SMTP)
5xx	erreur côté serveur (HTTP), ou erreur permanente (SMTP)

FIGURE 7.1 – Codes de résultats usuels des protocoles Internet ASCII NVT

6. https://developers.google.com/protocol-buffers/

7.2 Quelques protocoles applicatifs classiques

7.2.1 TELNET (informatif)

7.2.1.1 Fonctions et propriétés

TELNET permet à un utilisateur éloigné de s'annoncer (login) sur une machine et de l'utiliser comme s'il était connecté localement sur une console texte (démarrer des programmes, éditer, etc). Toutes les données entrées localement sont transmises à la machine éloignée. La transmission se fait avec le format NVT ASCII (*network virtual terminal*). Ce qui signifie que le format des caractères est ASCII (7 bits, voir `man 7 ascii` sur système UNIX). Le jeu complet est supporté et si ASCII n'est pas le format local, TELNET fait la conversion.

Les commandes qui forment de la **signalisation en-bande** sont précédées du caractère **IAC** (*Interpret As Command*), de valeur ASCII 255. Si un caractère ayant la valeur 255 doit être transmis tel, il est précédé du caractère IAC (**échappement**).

commande	**valeur décimale**	**usage**
SE	240	Fin d'une phase de négociation
NOP	241	Pas d'opération
DM	242	Redémarre le flux de sortie
BRK	243	Stoppe le flux de sortie
IP	244	Interruption du processus
AO	245	Abandon du flux de sortie
AYT	246	Es-tu là ?
EC	247	Effacement d'un caractère
EL	248	Effacement d'une ligne
GA	249	Continue
SB	250	Début d'une phase de négociation
WILL	251	L'expéditeur souhaite utiliser une option
WON'T	252	L'expéditeur souhaite désactiver une option
DO	253	L'expéditeur demande au destinataire d'activer l'option
DON'T	254	L'expéditeur demande au destinataire de désactiver une option
IAC	255	Interprète le prochain octet comme une commande

Un refus d'activation correspond à une désactivation. Une désactivation ne peur être refusée.

Exemples :

WILL x $\rightarrow$	l'expéditeur veut activer l'option x chez lui
DO x $\leftarrow$	le destinataire accepte que l'expéditeur active l'option x
WILL x $\rightarrow$	l'expéditeur veut activer l'option x chez lui
DONT x $\leftarrow$	le destinataire refuse que l'expéditeur active l'option x
DO x $\rightarrow$	l'expéditeur veut activer l'option x chez le destinataire
DONT x $\leftarrow$	le destinataire refuse d'activer l'option x

Quelques options : echo, suppress go ahead (pour full duplex), terminal type, window size...

Si une option n'est pas simplement activée ou désactivée mais nécessite un échange d'informations entre client et serveur, des sous-options sont utilisées (par exemple la transmission du type de terminal).

TELNET ne s'occupe pas de la partie d'identification de l'utilisateur qui est à gérer par le daemon appelé via un dialogue interactif avec l'utilisateur.

TELNET utilise le port 23.

7.2.1.2 Séquence des messages

L'échange se fait à l'aide d'une connexion TCP. Les caractères introduits par le client (provenant du clavier de l'utilisateur) sont souvent envoyés un à un ce qui est fort peu efficace (20 octets d'en-tête TCP, 20 octets d'en-tête IP et 14 octets d'entête Ethernet pour une charge utile de 1 octet – l'algorithme **NAGLE** est d'un précieux secours pour grouper les caractères !)

7.2.1.3 Evolution

rlogin (Remote Login) constitue une version simplifiée de TELNET qui ne fonctionne que entre machines UNIX et qui est aujourd'hui avantageusement – sécurité – remplacée par **SSH** pour la connexion distance ou le transfert de fichiers. SSH incorpore de l'identification d'utilisateur, éventuellement basé sur du **chiffrement asymétrique** (voir section 6.6.2.1 en page 106), voire même des connexions en mode graphique (**X11**) via un **tunnel**.

7.2.2 FTP (File Transfer Protocol)

7.2.2.1 Fonctions et propriétés

FTP est un protocole pour l'échange de fichiers[7] entre ordinateurs éloignés. Il constitue un sous-ensemble autonome puisqu'il comporte sa propre phase d'identification des utilisateurs. Il permet les fonctions suivantes :

- identification de l'utilisateur (en clair par défaut, SSL/TLS possible)
- sélection et navigation dans les répertoires
- création de répertoires et de fichiers
- transfert de fichier dans les deux sens (y compris reprise)
- effacement de fichiers

FTP connaît différents types de fichiers, différentes structures de fichier et différent mode de transmission. En pratique, seuls les cas suivants sont importants :

fichiers binaires les fichiers binaires sont transférés sans aucune conversion (compatibilité entre systèmes UNIX)

fichiers ASCII les délimiteurs de fin de ligne des fichiers ASCII sont au besoin convertis (UNIX : LF, Microsoft : CR/LF)

7.2.2.2 Ports

FTP a la particularité d'utiliser deux connexions TCP, une pour les commandes (sur le port 21 du serveur) et une pour les transferts (des ports dynamiques sont utilisés)

7. parmi les solutions modernes, citons le **sftp** de **SSH** et le **WebDAV** sur HTTPS

7.2.2.3 Messages

FTP utilise un format de commandes codées en ASCII.

Les principales commandes sont :

ABOR	abandon de la dernière commande et des transferts en cours
LIST	liste de fichier ou répertoires
PASS	mot de passe
PORT	adresse IP et port du client (mode actif)
QUIT	fin de session
RETR	transfert de fichier serveur → client (GET)
STOR	transfert de fichier client → serveur (PUT)
SYST	type de système (serveur)
TYPE	type de fichier (ASCII ou binaire)
USER	nom de l'utilisateur

Les réponses du serveur se composent d'un nombre à 3 chiffres et d'un texte. Ce genre de convention se retrouve p.ex. dans l'HTTP. En général les deux sont affichés. Les valeurs de nombres sont partiellement normalisées.

Exemple :

125	data connection already open ; transfer starting
200	command OK
245	can't open data connection

7.2.2.4 Séquence des messages

La connexion de commande est établie normalement (ouverture passive du serveur, ouverture active du client). Elle dure toute la session.

Une connexion de transfert est établie au besoin pour chaque transfert de fichier ou pour chaque liste de fichiers (ou de répertoires). Elle peut être initiée soit par le serveur (qui contacte le client, **mode actif**, ouverture active par le serveur), soit par le client (qui contacte le serveur, **mode passif**, ouverture active par le client). Ces deux modes ne sont pas toujours supportés, notamment en raison des **firewall**s.

En mode actif, le client fournit l'adresse IP et le port à utiliser (commande PORT) et le serveur s'y connecte en mode client. En mode passif (commande `PASV`), c'est le serveur qui fournit le tuple (adresse IP, port) et c'est le client qui s'y connecte, ce qui est en général compatible avec les **firewall**s du côté du réseau du client.

7.2.3 TFTP (Trivial File Transfer)

7.2.3.1 Fonctions et propriétés

Les fonctions de FTP sont parfois trop complexes, en particulier quand le transfert se fait à l'intérieur d'un réseau local. Ces réseaux ont des délais très courts et des taux d'erreurs très bas. TFTP n'est utilisé en principe que dans les LAN (démarrage de stations sans disque, téléchargement de configurations pour des équipements, ...). TFTP ne permet le transfert que

d'un fichier dans une direction ou l'autre et n'implémente pas d'identification de l'utilisateur. De plus, si le serveur ou le client ne supporte pas les extensions des RFC-2348 ou RFC-2349, la taille maximum transférable ne dépassera pas 32 MB – ce qui est en général largement suffisant.

TFTP utilise UDP, port 69

7.2.3.2 Séquence des messages

Le **protocole fiable** implémenté est un **IDLE REQUEST** sur la base d'UDP. Cela signifie que des datagrammes individuels UDP sont envoyés, et que chaque datagramme individuel doit être confirmé par le récepteur avant de passer au suivant, ce qui explique l'inefficacité du protocole, en particulier hors du réseau local (sensibilité aux délais).

Les messages individuels sont :

read request
write request
data block
acknowlegement

7.2.4 HTTP (HyperText Transfer Protocol)

Une introduction plus complète se trouve dans [10].

7.2.4.1 Fonctions et propriétés

Les fonctions de HTTP ont fortement évolué avec l'émergence du Web au cours des années 90. Aujourd'hui comme hier, HTTP est un protocole pour l'échange de données entre clients et serveurs Web. HTTP ne gère aucun contexte (souvent implémenté par des token ou cookies ou des paramètres d'URL ou de formulaires) et nécessite un protocole fiable dans la couche inférieure (typiquement TCP). On a toujours un échange simple d'une demande (request) et d'une réponse correspondante (response). L'initiative de l'échange est toujours décidé par le client (mode **PULL**[8]).

Les ressources accédées sont identifiées par un URI (uniform ressource identifier) dont les URL sont la variante couramment utilisée.

HTTP, dans ses versions récentes, prend en compte la présence éventuelle de proxy, de gateway et de tunnel. Certains paramètres du protocole et informations des entêtes de réponse du serveur servent par exemple à contrôler quelles informations peuvent être enregistrées dans un proxy-cache.

Ports : TCP, 80 (443 en mode chiffré TLS/SSL) – de plus les URLs permettent de spécifier un autre port si nécessaire.

Messages : Format NVT, avec entêtes.

8. le mode **PUSH** est très utilisé dans les applications Web modernes et permet au serveur d'informer le client via des notifications : cela est possible avec les **Web sockets** ou par polling (scrutation) de la part du client.

7.2.4.2 Isolation et sécurisation

Un **proxy** (avec ou sans cache) est un serveur intermédiaire qui peut garder des copies de documents et les fournir à des clients sans recours au serveur original. Ceci permet d'économiser les ressources du réseau, effectuer du filtrage (listes noires, anti-virus de contenu...) et limiter les échanges en couche 7 sans que le serveur distant ait accès aux couches inférieures IP du client. Un **reverse proxy** est un proxy protégeant un serveur Web, avec éventuellement une notion de répartition de charge (**load balancing**). Un **gateway** est un serveur Web permettant d'accéder à des ressources par d'autres protocoles (p.ex. convertir entre Internet général et un réseau **IoT** (*Internet of Things*) ; ou le protocole déjà obsolète **WAP** des téléphones mobiles). Un tunnel permet de faire passer la communication sur d'autres réseaux. La liaison entre un client et un serveur peut comporter plusieurs proxies, gateways ou tunnels et transiter par un reverse proxy avant d'atteindre le serveur HTTP proprement dit.

7.2.4.3 Les versions d'HTTP

HTTP 0.9 : Cette première version ne connaît qu'une seule commande. Elle permet à un client de demander le transfert d'un document au serveur (GET document.html). Le serveur répond par un transfert du document demandé. Pour chaque document transféré une connexion TCP est établie (partageant celle de contrôle contrairement à FTP) puis fermée à la fin du transfert.

HTTP 1.0 : Avec cette version le format **MIME** (text/html, image/gif...) des documents transférés est indiqué explicitement. Les indications de format sont faites de façon similaire à MIME comme le courrier électronique. Le client peut spécifier son format préféré et le serveur indiquer quel format est réellement utilisé. Deux nouvealles méthodes (commandes) sont ajoutées : HEAD (le client ne demande que les caractéristiques sans le document lui-même) et POST (le client peut fournir des informations à traiter par le serveur). POST permet par exemple l'envoi des données saisies dans un masque pour qu'elles soient traitées par le serveur Le serveur peut informer le client de façon plus détaillée en cas de problème. Une procédure d'identification de l'utilisateur est introduite. La possibilité de mettre en place des serveurs virtuels (noms de domaines) sous la même adresse IP est introduite.

HTTP 1.1 : Les principales améliorations sont les suivantes :

- de nouvelles commandes (PUT et DELETE) permettent de gérer le contenu du serveur via HTTP (p.ex. partage de fichier WebDAV, méthodes REST...).
- client et serveur s'interrogent concernant leurs caractéristiques (par exemple les formats supportés) (commande OPTIONS).
- plusieurs documents peuvent être transférés par une seule connexion TCP (**keep-alive**)
- le transfert d'un document peut être tronçonné.
- le transfert d'une partie de document est possible.
- le format de transfert d'un document peut être négocié entre le client et le serveur.
- l'identification des utilisateurs ne se fait plus seulement par un mot de passe transmis en clair (variantes multiples, par exemple par hachage).
- le contrôle du proxy par le serveur est plus fin : le serveur peut indiquer explicitement ce qui peut être mis dans le cache.
- des extensions permettent des notifications (PUSH).

Le protocole HTTP/2.0 (anciennement Google **SPDY**) ajoute la notion de multiplexage et de réordonnancement de requêtes afin d'augmenter la performance des échanges. Le protocole HTTP/3.0, quant à lui, peut être vu [9] comme HTTP/2.0 transporté en couche 4 non plus par

9. en fait une partie des améliorations apportées par SPDY est intégrée à QUIC, et une autre à HTTP

TCP mais par **QUIC** (implémentant le protocole fiable dans le navigateur sur UDP) – voir paragraphe 4.2.11.2 en page 90.

7.2.5 Courrier électronique : SMTP, POP, IMAP, MIME

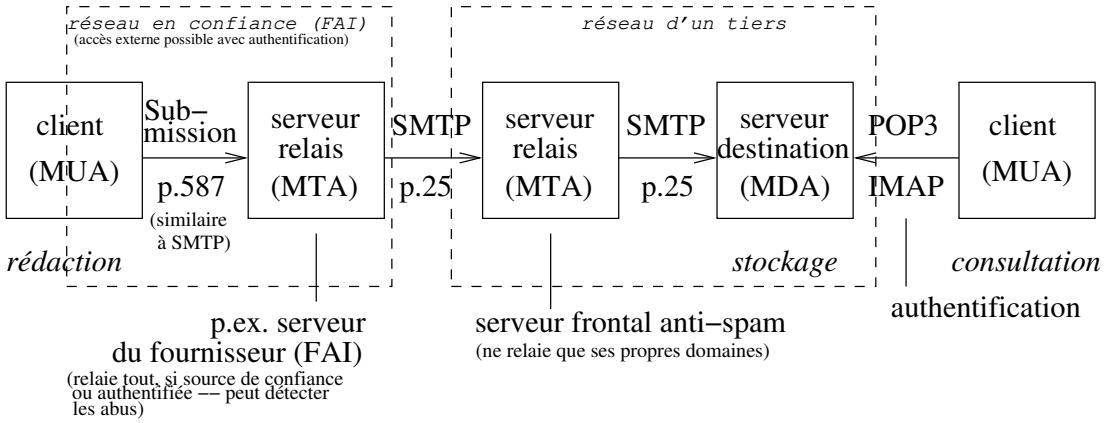


FIGURE 7.2 – Envoi d'un e-mail

7.2.5.1 SMTP (Simple Mail Transfer Protocol)

7.2.5.1.1 Fonctions et propriétés SMTP (RFC-821) permet le transfert de courrier électronique (e-mail) entre deux **MTA**s (*Message Transfer Agent*). La saisie des mails et leur routage ne sont pas des tâches de SMTP, mais celles respectivement du **MUA** (*Mail User Agent*) et du **MDA** (*Mail Delivery Agent*).

Le format de base des mails (RFC-822) est très simple : l'entête se compose ligne contenant un mot clé (To, From, Subject, Date...) suivi par deux-points, espace, puis une chaîne de caractère. Le corps est séparé de l'entête par une *ligne vide*.

Les MTA ajoutent une *enveloppe* autour du message. Elle ne contient que les adresses de l'expéditeur et du destinataire du message et sont transmises, dans le protocole SMTP, via les commandes `MAIL FROM:` et `RCPT TO:`. Le transfert du message (comprenant d'abord l'entête, une ligne vide, puis le corps du message), initié par la commande `DATA` est terminé par une ligne ne comprenant qu'un point [10].

On peut faire une analogie avec le courrier postal : la lettre est formée d'entêtes (expéditeur, destinataire) et d'un texte (le corps du message). C'est ce qui est transmis dans la session `DATA` et ce qui sera généralement présenté – sous forme résumée pour les entêtes – par le client mail (**MUA**). Mais une lettre est postée dans une enveloppe, qui contient un destinataire et souvent un expéditeur. C'est l'adresse de destinataire de l'enveloppe qui sert à l'acheminement de la lettre ainsi postée (le routage est fait sur le `RCPT TO:` – voir section 3.1.6.3.1 en page 65). L'adresse d'expéditeur de l'enveloppe sert à l'acheminement d'éventuelles erreurs (**NDN**, *Non Delivery Notification*).

7.2.5.1.2 MIME **MIME** (*Multipurpose Internet Mail Extensions*) a été défini afin de rendre plus flexible les données transférables par SMTP. A l'origine seul des textes ASCII pouvaient être envoyés. Grâce à MIME le corps d'un mail peut se composer de plusieurs parties ayant

10. une ligne de message commençant par un point doit être échappée en ajoutant un point et récursivement.

chacune un type différent. Chaque partie a son propre en-tête avec la description des types et sous-types de contenus.

Type de contenu	**Sous-type**	**Description**
text	plain richtext enriched	texte simple texte formaté extension de richtext
multipart	mixed parallel digest alternative	corps multiple à traiter en séquence corps multiple à traiter en parallèle condensé d'un mail plusieurs parties ayant la même sémantique (le client choisit)
message	rfc822 partial external-body	un autre mail (RFC-822 partie d'un mail pointeur sur un autre message
application	octet-stream postscript	contenu binaire quelconque programme postscript
image	jpeg gif	format ISO 10918 graphic interchange format
audio	basic	8 bits, ISDN, m-Law
video	mpeg	ISO 11172

Le protocole SMTP classique ne permet pas de passer tous les 8 bits des données (ni de supporter de longues lignes) : seul un sous-ensemble de l'ASCII 7 bit est possible (p.ex. NUL ASCII 0 est interdit).

Il faudra donc procéder à un encodage lors du transfert.

Deux encodages sont très souvent utilisés dans les protocoles Internet (p.ex. courrier électronique, HTTP) : **base64** et **quoted-printable** :

	base64	quoted-printable
principe	les octets (8 bits) sont encodés sur des sextets (6 bits, $2^6 = 64$) produisant un multiple de 24 bits, on signale les octets manquants par un ou deux =	échappement des caractères non représentables en ASCII 7 bits (et quelques exceptions) par une séquence d'échappement =XX (hexadécimal)
efficacité	en fait, trois octets deviennent 4 octets (débutant chacun par 2 bits à zéro) : augmente d'un tiers la taille des données (+ formatage)	bien meilleure efficacité, si les données originales ne sont pas composées en priorité de valeurs à échapper
utilisation	fichier binaires, images, etc	fichiers textuels accentués

7.2.5.1.3 Envoi SMTP La résolution des adresses e-mail en adresse IP se fait en résolvant la partie domaine (après le AT @) via **DNS** (*Domain Name Server*, champ **MX**, *Mail Exchanger*) (voir section 3.1.6.3.1 en page 65). Selon les entrées dans DNS le routage peut être direct ou indirect. Si le routage est direct une liaison TCP est établie entre le premier MTA ayant reçu le message et le MTA du destinataire et le mail est transféré par cette liaison. Si une entreprise utilise plusieurs MTA derrière un firewall, elle doit permettre un accès direct par SMTP à chaque machine concernée et le contrôle devient difficile. C'est pourquoi on utilise de plus en

plus souvent un routage indirect. Un MTA, éventuellement redondant, centralise tout le courrier entrant et sortant de l'entreprise et le redistribue sur les MTA internes.

Le port standard utilisé est 25 (TCP). Les clients (**MUA**) envoyaient au début aussi sur ce port. Entre temps, la possibilité existe d'utiliser plutôt le protocole Mail Submission, qui lui utilise le port 587 et est très similaire à SMTP : la différence principale est que l'authentification du client est obligatoire. Les FAIs filtrent de plus en plus le port 25 en provenance de clients pour éviter le **SPAM** et forcer le passage par le serveur de mail du fournisseur, permettant ainsi la détection d'abus.

7.2.5.1.4 Commandes Les commandes sont ici aussi en format NVT ASCII.

Commande	Description
HELO	Le client s'identifie auprès du serveur
MAIL FROM	Annonce de l'expéditeur du message
RCPT TO	Annonce du destinataire du message
DATA	Données du message (entête et corps)
QUIT	Fin de session
VRFY	Contrôle de l'adresse d'un destinataire
EXPN	Expansion d'une liste de distribution

Les réponses sont composées d'un numéro et d'un texte optionnel.

A l'établissement de la connexion le serveur envoie spontanément un message d'accueil (220 xxxxxxxx xxx). Le client s'annonce ensuite avec HELO (ou EHLO pour des fonctions étendues). Des extensions permettent d'authentifier le client (**SASL**) et/ou de chiffrer l'échange de données.

7.2.5.2 POP et IMAP

POP et IMAP sont deux protocoles d'accès aux boîtes-aux-lettres. Les nouveaux mails sont saisis à l'aide d'un logiciel local. Les mails sortants sont en général envoyés directement via SMTP (MTA du destinataire ou MTA centralisé). Les mails entrants doivent être enregistrés dans une boîte-aux-lettres car le PC du destinataire final n'est pas toujours accessible et ne tourne en général pas de **daemon** SMTP. Lors que celui-ci le souhaite, il s'annonce auprès de sa boîte aux lettres et relève le courrier. POP et IMAP sont prévu pour cette interface entre client mail et boîte-aux-lettres.

POP fonctionne selon un mode hors-ligne (*offline*). A l'établissement d'une connexion avec la boîte-aux-lettres unique, tous les mails pendants sont envoyés au client et effacés du serveur (en général).

Avec IMAP, on a le choix de la ou les boîtes-aux-lettres (une hiérarchie est présentée), et les mails sont toujours conservés sur le serveur : ils ne sont transférés au client mail qu'en cas de besoin (lecture totale ou partielle, cache). Ceci représente un grand avantage lorsqu'on lit ses mails de plusieurs endroits (ordinateur, téléphone portable). IMAP prévoit de plus des fonctions nécessaires à la gestion de répertoires de mails sur le serveur et propose des fonctions avancées de notification pour informer le client de l'arrivée de nouveaux messages.

7.2.5.3 Adresses e-mail à vie ou jetables

Pourquoi ne pas créer une adresse à vie, indépendante de l'employeur et redirigeant tout le courrier reçu sur votre e-mail actuel, auprès d'un FAI de confiance ou d'une association ?

Il peut aussi être utile de gérer une adresse e-mail par type d'activité, voire même d'utiliser des adresses dites jetables [11], utilisables pour s'abonner à des services sans risquer de recevoir encore plus de **SPAM**.

7.2.6 Echange de messages

En plus du courrier électronique (voir section 7.2.5 en page 121), plutôt utilisé entre humains, les applications veulent pouvoir échanger des messages. De nombreux protocoles ont été définis dans ce but, que cela soit pour le format des messages (ou télégrammes) échangés ou les protocoles de transport.

Les critères pour la sélection d'un format [12] sont liés à la taille des messages et aux équipements qui les traitent, même si l'on s'achemine de plus en plus, lorsque c'est possible, vers des formats textes ouverts, par exemples basés JSON ou XML. Quant au transport, il dépend du type de service souhaité : livraison en temps réel [13] ou différée (**store and forward**, en particulier pour les mobiles économes en énergie).

Parmi les protocoles existants, la plupart sont basés sur un mécanisme de publication et d'abonnement ; citons les plus populaires :

- **XMPP** : XML transporté par TCP simple, voire HTTP, avec ou sans chiffrement, pour des applications de messages humains et/ou machines, principalement interactives ; centralisé
- **IRC** : protocole NVT classique sur TCP, éventuellement avec chiffrement, pour des applications de messages humains ou inter-machines (p.ex. réseaux d'attaque **command-and-control**), uniquement interactives ; partiellement décentralisé
- **MQTT** : supporte le non interactif (*store and forward*)
- beaucoup de services utilisant JSON transporté via REST sur HTTPS

7.2.7 Protocoles distribués et décentralisés

La majorité des protocoles Internet restent des protocoles de type **architecture client/serveur**. Même le **cloud** est en général, du point de vue du client, un service centralisé.

Toutefois, des protocoles distribués décentralisés existent également : citons par exemple l'échange de fichier **P2P**, le logiciel **peertube**, qui, au sein d'un navigateur permet de voir des vidéos sans surcharger le serveur central, ou encore le système de fichiers **ipfs** qui permet de stocker des fichiers confidentiels à plusieurs exemplaires sur des ordinateurs de tiers, et de stocker si nécessaire des références sur la blockchain distribuée Ethereum (*Ethereum Name Service*).

Si ipfs a ajouté le stockage quasi-persistent, la distribution des données [14] de bouts de fichiers hachés), **IPLD** semble être la concrétisation d'adresses universelles pour les associations. Enfin, le W3C va mettre en commun tous les protocoles **P2P** avec le préfixe `dweb://` Certains navigateurs le supportent déjà.

Un jour peut-être, le web deviendra décentralisé et à haute fiabilité : le **web3**.

11. https://jetable.org
12. voir section 7.1.3
13. une communication directe P2P via socket ou WebRTC est toujours possible, même si un modèle client/serveur centralisé a l'avantage de faciliter le déploiement
14. un URL est un emplacement et un chemin ipfs est une donnée où qu'elle se trouve ; **IPNS** permettant le routage par résolution de nom

Références et bibliographie

[1] Cisco, *Symboles d'équipements réseau*, http://www.cisco.com/web/about/ac50/ac47/2.html, consulté le 2019-08-28

[2] Claude Servin, *Réseaux et télécoms*, 2e édition, Dunod, ISBN 2-10-049148-2

[3] Fred Halsall, *Data Communications, Computer Networks and Open Systems* (Fourth Edition), Addison-Wesley 1996, ISBN 0-201-42293-x.

[4] Andrew S. Tanenbaum, *RESEAUX : Architectures, protocoles, applications*, InterEditions 1990, ISBN 2-7296-0301-8, (Traduction de Computer Networks, Prentice-Hall 1989).

[5] Antoine Delley, *Téléinformatique*, Dunod Informatique 1987, ISBN 2-04-016907-5.

[6] *Information Technology Networks*, ISBN 2-940156-26-3, 1ère édition, HES-SO Fribourg

[7] W. Richard Stevens, *TCP/IP Illustrated volume 1 : The Protocols*, Addison-Wesley, ISBN 0-201-63346-9

[8] Christian Huitema, *Le routage dans l'Internet*, Eyrolles, 10/1994, 418p, ISBN 2-212-08902-3 (10/1994)

[9] Shon Harris, *All-in-one CISSP exam guide*, 4th Edition, Mc Graw-Hill, ISBN 978-0-07-149786-2

[10] Marc Schaefer, *Applications Web*, ISBN 978-2-9403-8705-2, 2023

[11] Marc Schaefer, *Gitlab des cours* et notamment la bibliographie téléinformatique.

Lexique

Voici un lexique de quelques termes liés aux réseaux. N'hésitez pas à consulter également l'index des concepts (voir 7.2.7 en page 129). Le RFC-1392 aussi contient une liste des termes les plus courants.

bande passante Désigne la plage de fréquence, en Hertz [Hz], qui permet de transmettre le signal. Formellement, la bande passante (anglais : *bandwidth*) peut se définir comme la plage de fréquence dans laquelle la puissance du signal n'est pas diminuée de plus de 3 dB par rapport au maximum – soit moins de la moitié.

Attention : ne pas confondre avec le **débit binaire** (voir section 1.4.2 en page 30) obtenu après choix de la modulation et qui dépend de la bande passante choisie ainsi que nombre d'états possibles en fonction du **rapport signal-sur-bruit**. Cette confusion est malheureusement quasi systématique.

bourrage Rajouter des données pour atteindre une taille minimale : par exemple, Ethernet et GBit Ethernet ont des tailles de trames minimales de 64 : des octets supplémentaires (à ignorer) sont ajoutés par la couche 2 en cas de nécessité. De même, en couche 3, un entête IP doit toujours comporter un nombre entier de mots de 32 bits (4 octets) : certaines options IP étant plus courtes, du bourrage sera également effectué.

décibel Unité très pratique de mesure, par exemple pour la puissance d'un signal. Comme elle est basé sur les logarithmes, elle est pratique à utiliser : par exemple, lors d'un calcul de liaison, les gains successifs s'additionnent lorsqu'ils sont exprimés en **dB**, alors qu'ils se multiplient en puissance (Watt, [W]).

De nombreux *types* d'unité de décibel existent, qui permettent de spécifier la référence. Par exemple, 20 **dBm** correspondent à 10^2 mW, soit 100 mW, car la référence de 0 dBm est à 1mW. Sans spécification d'unité, une valeur en décibel est simplement un rapport, un multiplicateur sans unité.

symétrique Concept très surchargé, signifiant généralement quelque chose qui a une symétrie, qui est la même chose vu de deux points de vue différents. Contraire d'asymétrique : qui n'est pas symétrique. Dans le contexte des réseaux, voici quelques exemples :

- l'**architecture client/serveur** est asymétrique : le client pose les questions, le serveur y répond (**PULL**)
- une **paire symétrique** est une paire avec deux fils de même caractéristiques, qui sont torsadés l'un sur l'autre et donc affecté par le bruit électromagnétique de manière similaire : une lecture différentielle des deux fils annule en grande partie le bruit ; exemple : les 4 paires symétriques d'un câble Ethernet
- une liaison symétrique a le même débit montant et descendant, une liaison asymétrique des débits différents (le cas le plus courant pour des liaisons Internet vers des clients finaux : **ADSL**, *Asymmetric Digital Subscriber Line*
- un chemin (la route) dans le réseau peut être asymétrique : on ne passe pas par les mêmes routeurs, ce qui peut poser des problèmes d'estimation du délai
- le chiffrement peut être symétrique (même clé pour chiffrer/déchiffrer) ou asymétrique (2 clés différentes)

télécommunication Du grec *télès* (distant) : souvent au pluriel, désigne l'ensemble des moyens de communication à distance.

téléinformatique Mot-valise désignant l'informatique faisant appel aux moyen de télécommunications.

télématique Mot peu courant aujourd'hui désignant l'ensemble des services informatiques fournis à travers un réseau de télécommunications (néologisme créé en 1978 par Alain Minc et Simon Nora).

trame Un **PDU** de couche 2. En général, on parle de **paquets** aux niveaux 3-7 du modèle OSI, et de trame en couche 2. On parle aussi de datagramme dans le modèle IP, en couche 3.

xDSL Se dit des technologies d'accès au réseau (du dernier kilomètre) basées sur le DSL (*Digital Subscriber Line*, ligne numérique d'abonné, soit la ligne téléphonique classique (**POTS**, *Plain Old Telephone System*) utilisée au-delà de la **bande passante** téléphonique), comme par exemple l'**ADSL** (débit asymétrique, destiné aux usagers résidentiels), le **VDSL** (similaire à l'ADSL, meilleur débit et distances) et les **SDSL** (débit symétrique, destiné aux entreprises).

Index des concepts

Table des figures

Table des matières

ISBN 978-2-940387-07-6