Réseaux

Emmanuel Baccelli, Thomas Heide Clausen

Réseaux

E. Baccelli[1] and T. Clausen[2]

[1]INRIA
[2]Ecole Polytechnique

Chapitre 1

Réseaux

1.1 Cours

Qu'est-ce que la communication ? Vaste question. Ce chapitre ne prétend pas y répondre dans son intégralité, mais présente les principes qui régissent de nos jours la communication entre ordinateurs, et notamment Internet. Pour mieux en saisir les concepts, ce chapitre commence par une analyse sommaire de certaines formes de communications entre êtres humains, qui se prête facilement à une analogie avec la manière dont les ordinateurs communiquent entre eux.

1.1.1 Communication entre êtres humains

Une des formes de communication les plus immédiates est la communication orale, qui peut prendre diverses formes, du chant au discours, dans des langues variées. Cependant, toutes ces formes ont en commun le fait d'être un moyen pour véhiculer des idées, d'un émetteur à un récepteur : une mère chantant une berceuse à son enfant lui communique des idées combinant sécurité et calme, tandis qu'un homme politique prononçant un discours devant ses électeurs potentiels leur communique des idées alliant "je comprends la situation" et "je pourrais bien vous représenter".

Faute de pouvoir employer la télépathie, on a donc souvent recours à l'oral pour communiquer une idée entre êtres humains. L'esprit de l'émetteur de l'idée la formule en mots et en phrases, qui sont transmis au récepteur qui les

comprend et peut ainsi se représenter l'idée dans son propre esprit. Pour être communiqués par la parole, les mots et les phrases doivent être traduits en ondes sonores produites par les cordes vocales de l'émetteur. Ces ondes sont le *support physique* de la communication et se propagent jusqu'aux oreilles du récepteur. Les vibrations induites des tympans de ce dernier sont retraduites en mots et en phrases (voir figure 1.1 qui retrace cette décomposition en huit étapes). Bien sûr, pour que la communication marche, il faut que les mots soient compréhensibles, dans une langue pré-établie, et prononcés de manière suffisamment intelligible pour être entendus correctement.

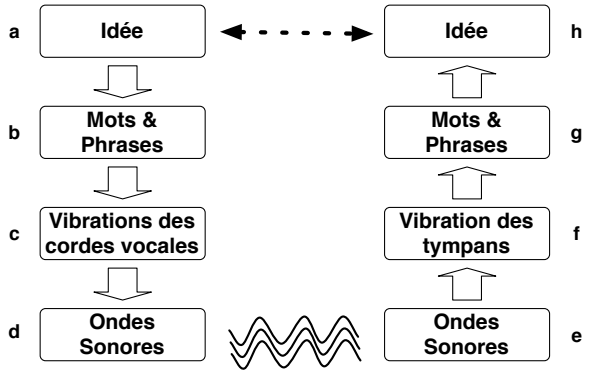


FIGURE 1.1 – Communication orale entre êtres humains.

Une autre forme de communication souvent utilisée est la communication écrite. Par rapport à l'oral, le support physique change (ce ne sont plus des ondes sonores mais un support écrit), mais l'émetteur formule toujours ses idées en mots et en phrases qui sont transmis au récepteur qui les comprend, puis saisit les idées. Ce mode de fonctionnement commun permet par exemple à un journaliste de servir d'*intermédiaire*, et de pouvoir retranscrire ce qu'a déclaré une actrice de cinéma lors de sa dernière interview (voir figure 1.2 qui retrace cette transmission). Des lecteurs pourront ultérieurement en prendre connaissance, pourvu bien sûr que l'écriture soit lisible, et dans une langue connue.

Comme nous allons le voir par la suite, la communication entre ordinateurs fonctionne grâce à des mécanismes équivalents à ceux décrits jusqu'ici pour la communication entre êtres humains.

À Retenir :

– La communication est un moyen, et non un but.

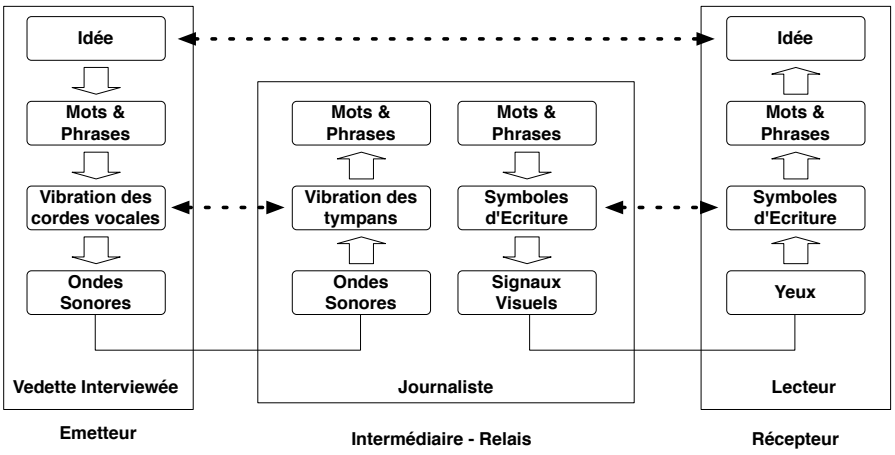


FIGURE 1.2 – Lecture d'une retranscription écrite d'une interview orale.

– Les idées communiquées d'un esprit à un autre sont l'essentiel, les moyens de communication sont variables et secondaires.
– Des mécanismes communs à différentes formes de communication permettent de relayer une idée à travers des intermédiaires, au moyen de supports qui peuvent être hétérogènes.

1.1.2 Communication entre ordinateurs, réseaux d'ordinateurs

Un ordinateur peut manipuler des quantités d'informations diverses, pourvu que ces informations soient représentées sous forme binaire, c'est-à-dire écrites au moyen d'un alphabet simplifié contenant seulement deux lettres, qu'il est coutume de représenter par les chiffres 0 et 1. On appelle une telle lettre un *bit*, et dans cet alphabet, les mots et les phrases représentant l'information s'écrivent donc sous forme de suites de bits, comme par exemple 0001011100010. Dans le suite de ce chapitre, on appellera ces informations binaires des *données*.

Comme les humains, les ordinateurs sont capables de communiquer entre eux quand ils sont reliés les uns aux autres, par exemple via Internet. Au travers de programmes appelés *applications*, les internautes peuvent transmettre d'un ordinateur à l'autre des amas de bits cohérents appelés *fichiers* représentant par exemple du texte, du son ou de l'image. Les applications

sont la partie 'visible' d'Internet, que tout le monde connaît et utilise tous les jours. Le couriel, la navigation web, ou le chat sont des exemples d'applications courantes.

Les fichiers transmis peuvent être reçus quasiment instantanément à des dizaines de milliers de kilomètres de là, comme par magie. Comment cela marche-t-il ? Le fonctionnement Internet est simple, comme nous allons le voir dans la suite de ce chapitre : il s'agit en fait de décomposer un problème complexe en plusieurs sous-problèmes plus limités et plus faciles à résoudre. Pour en saisir les principes, on peut reprendre l'analogie avec la communication écrite entre humains, sous la forme épistolaire, étape par étape.

Étape 1 : L'équivalent de savoir écrire et lire les lettres de l'alphabet. Pour les ordinateurs, sachant que les informations à transmettre sont représentées par des bits (les chiffres 0 ou 1), le premier pas est d'être capable de transférer un 0 ou un 1 au moyen d'un support physique reliant deux ordinateurs (le plus souvent par câble ou par radio). Ceci est analogue à savoir écrire et lire les lettres de l'alphabet pour communiquer par écrit entre humains.

Étape 2 : L'équivalent de savoir écrire et lire des mots et des phrases. Pour un ordinateur, il s'agit de coordonner le transfert groupé de suites de bits d'un bout à l'autre du câble ou du lien radio. On appelle de tels groupements des *paquets*, analogues à des mots et des phrases en langage humain.

Étape 3 : L'équivalent de savoir fournir un service postal pour acheminer des lettres à bon port. Pour un ordinateur qui veut joindre un autre ordinateur distant il faut gérer le transfert de paquets à travers des ordinateurs intermédiaires, interconnectés par une série de câbles et/ou liens radio qui mènent le paquet jusqu'à destination. On appelle cette interconnexion un réseau d'ordinateurs, un réseau de donnée ou plus simplement : un *réseau*. Le plus vaste et le plus connu d'entre eux est Internet, qui, par sa taille, est sans doute la plus grande construction jamais réalisée par l'homme.

Étape 4 : L'équivalent de savoir adapter les envois au format lettre du service postal disponible. Par exemple, pour pouvoir transférer des fichiers entiers d'un ordinateur à un autre à travers le réseau, il faut pouvoir fragmenter l'information d'un fichier en plusieurs paquets (un fichier est souvent trop gros pour tenir dans un seul paquet). Il faut également assurer la fiabilité du transfert de chaque paquet jusqu'à destination, où le fichier sera alors réassemblé.

Étape 5 : L'équivalent de la communication par lettres. Pour un ordinateur il s'agit de permettre aux applications d'utiliser Internet pour envoyer

des données vers d'autres applications sur d'autres ordinateurs, à travers le réseau.

Organisation en pile

Les étapes listées précédemment correspondent à une décomposition type, appelée organisation en pile, qui régit les réseaux d'ordinateurs en général et Internet en particulier. Cette décomposition est similaire à l'empilement décrit précédemment pour la communication entre humains (voir Section 1.1.1).

La communication entre ordinateurs, quant à elle, utilise la pile décrite figure 1.3 illustrant l'empilement des couches présentes sur chaque ordinateur du réseau, ainsi que leurs interactions. Chaque couche est constituée d'un ensemble de programmes dédiés à fournir les fonctionnalités listées dans l'une des 5 étapes décrites précédemment. On appelle les programmes dédiés à faire fonctionner le réseau *des protocoles*. Ce chapitre introduit les mécanismes fondamentaux à l'œuvre dans les protocoles, ainsi que l'architecture *en couche* qui les gouverne ensemble :

Couche 1. Appelée couche physique cette couche est constituée des protocoles responsables du transfert individuel d'un bit (0 ou 1) à travers un support physique (généralement un câble ou un lien radio). Nous verrons des exemples simples de mécanismes à l'œuvre au sein de cette couche à la section 1.1.3.

Couche 2. Appelée couche lien, cette couche est constituée des protocoles responsables de la coordination du transfert de paquets à travers un support physique, d'identifier les ordinateurs directement connectés à ce support, et de gérer le "temps de parole" de chacun sur ce support. Des exemples typiques de protocoles de cette couche sont le Wifi ou l'Ethernet. La section 1.1.4 détaillera les techniques essentielles utilisées à la couche lien.

Couche 3. Appelée couche réseau, cette couche est constituée des protocoles responsables de l'aiguillage de chaque paquet vers sa destination, à chaque embranchement entre différents supports physiques rencontrés au cours du périple de ce paquet à travers le réseau. Les protocoles de la couche réseau doivent également identifier les ordinateurs connectés sur le réseau. Le protocole le plus connu de cette couche est le protocole IP (Internet Protocol). La section présentera plus en détail les mécanismes fondamentaux utilisés par la couche transport.

Couche 4. Appelée couche transport, cette couche est constituée des protocoles responsables de l'empaquetage des données à transmettre, ainsi que de la coordination entre l'envoyeur et le destinataire des paquets, pour assurer la fiabilité de leur transport de bout en bout. La couche transport doit également identifier les applications en cours d'éxecution qui utilisent le réseau. Le protocole le plus connu de cette couche est TCP (Transmission Control Protocol). La section présentera plus en détail les mécanismes de base utilisés par la couche transport.

Couche 5. Appelée couche application, cette couche est constituée des programmes appelés *applications* qui utilisent le réseau. Tout comme la transmission d'idées est le but de la communication entre humains (voir section 1.1.1), le but de la communication entre ordinateurs est la transmission de données entre applications. Les programmes de cette couche ne sont pas dédiés à faire fonctionner le réseau, et ne sont donc pas abordés en détails dans ce chapitre.

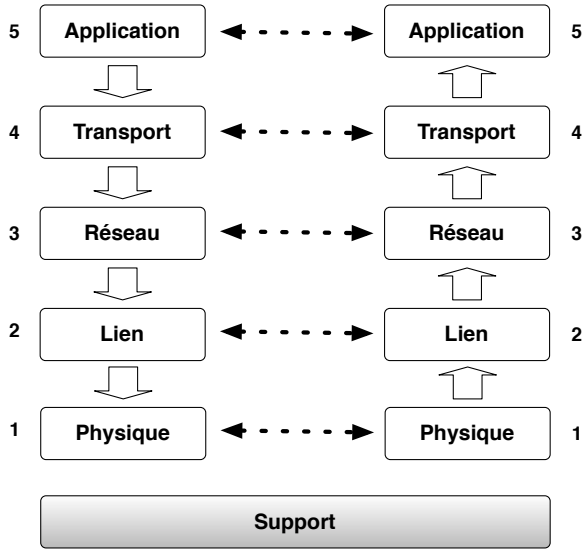


FIGURE 1.3 – Communication entre deux ordinateurs : organisation en pile présente sur chaque ordinateur. Chaque couche empilée communique avec son homologue (flèches horizontales) pour fournir des services à la couche immédiatement supérieure, en utilisant les services fournis par la couche immédiatement inférieure.

Chaque couche peut correspondre avec son homologue (flèches horizontales)

située sur un autre ordinateur, à travers une "boîte aux lettres" fournie par la couche du dessous, schématisé par des fonctions interfaces simples, comme *recevoir*() et *envoyer*(). Quand sur un ordinateur la couche n reçoit un paquet P à transmettre de la part de la couche du dessus $n+1$ à son homologue sur un autre ordinateur, elle encapsule tel-quel ce paquet, dans un paquet plus grand P' avec en en-tête (ou *header* en anglais) les informations nécessaires, spécifiques au bon fonctionnement de la couche n, pour l'acheminement du paquet vers la couche n de l'ordinateur destinataire, soit : $P' = [header(n) : P]$.

Autrement dit : les paquets à transmettre pour le compte de la couche $n+1$ sont *tels quels* le contenu des paquets transmis par la couche n. Cette dernière ajoute un en-tête contenant des informations de contrôle qui sont utilisées par la couche n, comme par exemple l'adresse de l'ordinateur destinataire, et certaines autres à l'aide desquelles des services élaborés peuvent être fournis par la couche n. La figure 1.4 illustre ce principe pour $n = 5$.

Le paquet poursuivant son parcours, la couche n sollicite ensuite la couche $n-1$ pour envoyer P'. Cette dernière encapsule P' selon le même principe, dans un paquet plus grand $P'' = [header(n-1) : header(n) : P]$. Et ainsi de suite jusqu'à la couche 1 (la couche physique) qui transmet le paquet final $[header(1) : ... : header(n-1) : header(n) : P]$ sous forme de suite de 0 et de 1 à travers un support physique.

Le paquet est réceptionné par la couche 1 de l'ordinateur destinataire, qui consulte l'en-tête la concernant, à savoir $header(1)$), et si estimé correct, transmet le contenu du paquet de son point de vue, soit $[header(2) : ... : header(n-1) : header(n) : P]$, à la couche 2 de l'ordinateur destinataire. Cette dernière consulte l'en-tête la concernant, et si estimé correct, transmet le contenu du paquet de son point de vue (omettant donc $header(2)$) à la couche 3 et ainsi de suite jusqu'à la couche $n+1$ qui reçoit donc finalement le paquet P qui lui a été envoyé.

Les informations de contrôle contenues dans l'en-tête utilisé par la couche n contiennent en général de nombreux renseignements utiles pour les couches n homologues d'autres ordinateurs. Cet en-tête contient entre autres choses l'identité de la destination du paquet, du point de vue de la couche n. Chaque couche utilise un système d'identification spécifique : un certain format d'adresse adapté aux tâches particulières que doit accomplir cette couche. En conséquence, quand la couche $n-1$ reçoit un paquet $[header(n) : P]$ à transmettre pour le compte de la couche n, l'identité de la destination du paquet du point de vue de la couche $n-1$ doit être déduite de l'adresse

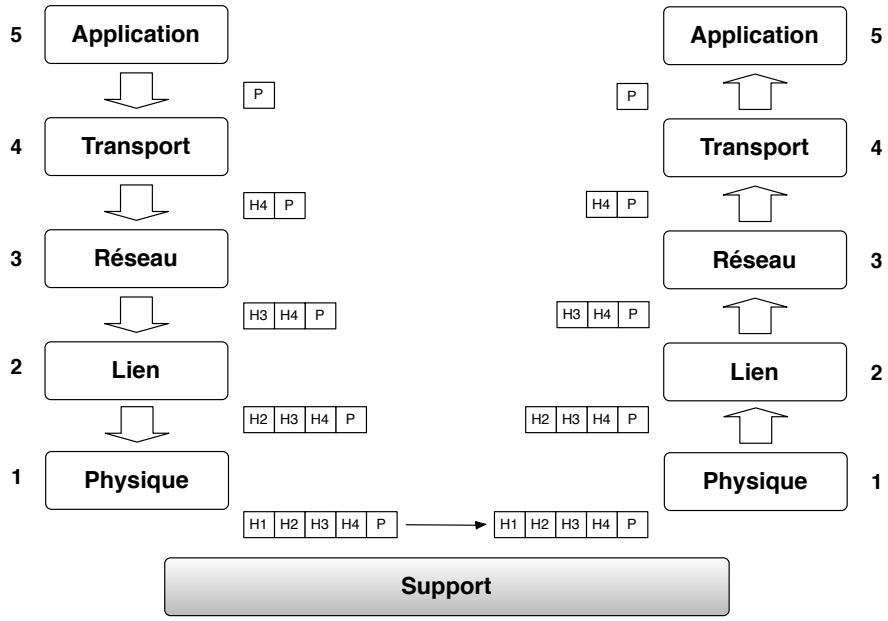


FIGURE 1.4 – Interactions verticales entre les couches. En-têtes imbriqués.

contenue dans l'en-tête de la couche supérieure $header(n)$. L'identité déduite sera alors renseignée dans $header(n-1)$. Ce procédé s'appelle la *résolution d'adresse*.

Les informations de contrôle contenues dans l'en-tête utilisé par la couche n ainsi que leur traitement par la couche n homologue sur l'ordinateur destinataire (interactions horizontales dans la figure 1.3) sont en général complexes, au contraire de l'interaction entre la couche n et les couche $n+1$ et $n-1$ (interactions verticales dans la figure 1.3), qui se résume simplement aux fonctions *recevoir*() et *envoyer*(). Ceci permet notamment de totalement changer les mécanismes d'une couche sur un ordinateur sans avoir à changer les mécanismes des autres couches, tant que les interfaces *recevoir*() et *envoyer*() avec les couches immédiatement supérieures et inférieures sont conservées. L'organisation en pile permet essentiellement aux mécanismes internes d'une couche d'être "agnostique" concernant les mécanismes internes des autres couches.

La suite de ce chapitre rentre plus en détail dans le fonctionnement de chaque couche, ainsi que de quelques protocoles clefs. Il existe d'autres modèles d'organisation en couche que celui décrit ci-dessus. Le plus connu d'entre eux

est le modèle OSI (Open Systems Interconnection) qui définit sept couches au lieu des cinq présentées ci-dessus. Malgrès cette différence, le modèle OSI fonctionne selon le même principe générique d'organisation en pile.

À Retenir :
- Les ordinateurs communiquent entre eux de manière similaire aux êtres humains entre eux, au moyen d'un alphabet simplifié et de langues que l'on nomme protocoles.
- Les protocoles sont organisés en couches empilées les une sur les autres, présentes sur chaque ordinateur du réseau.
- Sur un ordinateur, chaque couche interagit de manière complexe avec la couche de même niveau (son homologue) sur l'ordinateur avec lequel on communique.
- Sur un ordinateur, chaque couche interagit de manière simple avec les couches qui lui sont directement inférieures et directement supérieure.

1.1.3 La couche physique

Comme on l'a vu dans la section 1.1.2, la tâche accomplie par les protocoles de cette couche est le transfert individuel d'un 0 ou d'un 1, d'un bout à l'autre d'un support physique. Plusieurs types de supports physiques sont utilisés pour connecter les ordinateurs entre eux : des câbles métalliques véhiculant des électrons, des câbles optiques véhiculant des photons, l'air véhiculant des ondes radios sont des exemples de supports physiques utilisés de nos jours. Pour chacun de ces supports il existe des protocoles spécialisés dans le transfert individuel d'un 0 ou d'un 1 d'un bout à l'autre du support.

Pour saisir le principe de base de ces protocoles, on peut se rappeler de systèmes de communication très anciens comme par exemple celui utilisé lors de l'élection du pape, qui date du Moyen-Age. L'élection se fait à huit-clos dans une chapelle, où sont enfermés les électeurs (les cardinaux), qui n'ont le droit de communiquer avec le reste du monde qu'à travers la cheminée de la chapelle jusqu'à ce qu'un nouveau pape soit élu : après chaque scrutin, les cardinaux communiquent les résultats par une fumée noire (vote non-concluant, l'équivalent d'un 0) ou par une fumée blanche (vote concluant, l'équivalent d'un 1). En voyant la couleur de la fumée, les observateurs extérieurs peuvent alors comprendre le message élémentaire envoyé par les cardinaux : un nouveau pape est-il élu, oui ou non, 0 ou 1.

Les protocoles modernes à l'œuvre dans la couche physique fonctionnent sur une base similaire. À la place de signaux de fumée observables à l'échelle macroscopique, on utilise des signaux observables à l'échelle microscopique, à base d'ondes électromagnétiques. À la place des variations de couleurs (blanc ou noir) pour coder l'information binaire, on utilise des variations de longueurs d'ondes, de phase ou d'intensité du signal etc.

Dans le cas de la communication par des signaux de fumée lors de l'élection du pape, le feu est traditionnellement un feu de paille. Celle-ci est mouillée quand il faut produire une fumée blanche. Depuis quelques années, des fumigènes sont utilisés en complément, pour éviter les confusions qui peuvent être causées par une fumée trop grise, ou une fumée pas assez visible par mauvais temps.

De la même manière, les protocoles modernes à l'œuvre dans la couche physique utilisent des compléments sophistiqués pour renforcer la clarté du signal, et le rendre plus résistant aux erreurs d'interprétation à la réception. Une des techniques utilisées, par exemple, est de transmettre des séquences pré-établies à l'avance, par exemple transmettre la suite 10110111000 au lieu de transmettre simplement 1, et transmettre 01001000111 au lieu de transmettre 0 (illustré par la figure 1.5). À la réception, on sait qu'on ne devrait recevoir qu'une suite de séquences complètes et correctes, et cela permet donc de deviner l'information d'origine même quand la transmission d'une partie d'un séquence est brouillée. Cette technique est notamment utilisée dans les communication sans-fil (notamment au sein de la couche physique utilisée avec le protocole Wifi), qui doivent souvent composer avec des signaux très brouillés par les interférences, les obstacles etc. Le coût de cette résistance aux erreurs de transmission est donc un certain nombre de transmissions supplémentaires effectuées a priori, en amont d'erreurs potentielles.

1.1.4 La couche lien

Comme on l'a vu dans la section 1.1.2, les protocoles de cette couche sont responsables de la coordination du transfert de paquets à travers le support physique, d'identifier les ordinateurs directement connectés à ce support, et de gérer le "temps de parole" de chacun sur le support. Dépendant du support physique, les protocoles utilisés diffèrent, notamment en ce qui concerne la gestion du temps de parole de chacun sur le support.

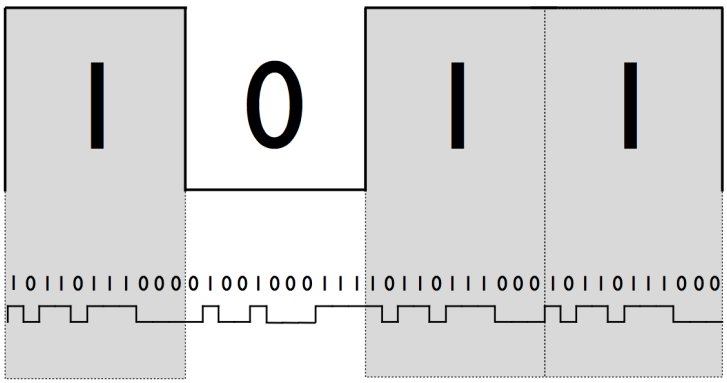


FIGURE 1.5 – Codage imbriqué pour renforcer le signal. Transmission de séquences de bits prédédinies (en bas), interprétées chacune comme un bit unique (en haut), à la réception.

Pour saisir les concepts de base de ces protocoles, un exemple pionnier datant des années 1970 est éloquent : ALOHAnet, un réseau d'ordinateurs utilisant des communications sans-fils pour connecter des ordinateurs dispersés sur les îles de l'archipel d'Hawaï à un ordinateur central situé sur l'une d'entre elles. La contrainte principale à cette époque était que tous les ordinateurs du réseau ALOHAnet devaient utiliser l'unique fréquence radio disponible pour communiquer avec l'ordinateur central. Cette contrainte crée des situations où plusieurs ordinateurs pourraient simultanément tenter d'envoyer chacun un paquet à l'ordinateur central, et sans le savoir, brouilleraient mutuellement leurs messages qui deviendraient incompréhensibles pour l'ordinateur central. On appelle ce brouillage mutuel une *collision* entre paquets, similaire à la situation où deux personnes parlent en même temps à une troisième qui, de ce fait, ne comprend rien.

Une solution pour gérer les collisions est de figer un ordre tournant que les ordinateurs doivent respecter pour que chacun puisse transmettre à son tour pendant en certain temps. Cependant cette solution centralisée a des inconvénients. D'une part si l'on rajoute ou enlève des ordinateurs, il faut tout reprogrammer avec un nouvel ordre à respecter. D'autre part, avec cet ordre systématique, on brime un ordinateur qui a soudainement beaucoup à transmettre si pendant ce temps là les autres n'ont rien à transmettre.

Une autre solution a donc été utilisé pour gérer les collisions de manière distribuée et plus flexible : le *protocole ALOHA*. Son mécanisme est simple :

chaque ordinateur est identifié par une adresse représentée sous forme d'une suite de k bits (k étant fixé à l'avance par convention). Dans chaque paquet à transmettre, l'ordinateur rajoute en en-tête son adresse ainsi que celle de la destination. Dès qu'un ordinateur a un paquet à transmettre, il l'envoie immédiatement, et attend un acquittement de la part de la destination lui confirmant que le paquet a bien été reçu. Si l'acquittement n'est pas reçu avant un temps d'attente maximum (fixé à l'avance), on estime que le paquet a été victime d'une collision. Dans ce cas, l'ordinateur attend un certain temps avant de transmettre le paquet de nouveau, la valeur de ce temps d'attente étant choisi aléatoirement pour réduire les chances de nouvelles collisions (voir figure 1.6). Si un paquet subit trop d'échecs de transmissions, le protocole abandonne.

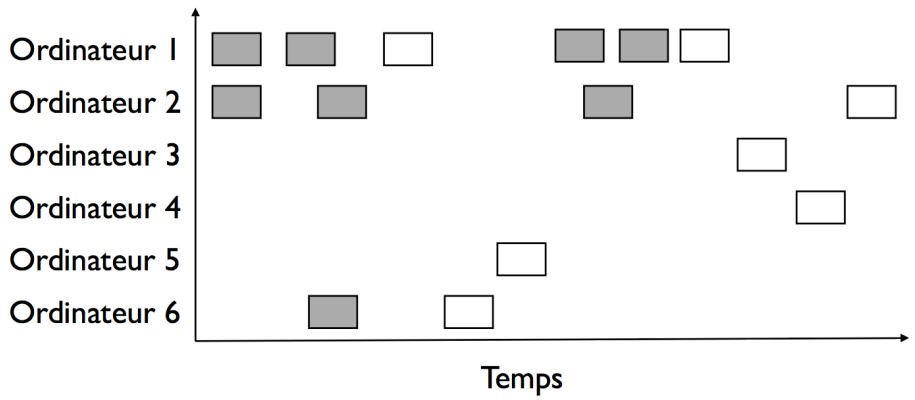


FIGURE 1.6 – Protocole ALOHA. Paquets victimes de collisions en gris. Paquets correctement transmis en blanc. Le temps d'attente entre un paquet gris et le prochain paquet sur le même ligne est aléatoire.

De nos jours, les protocoles utilisés à la couche lien sont le plus souvent *Ethernet* par câble, ou *WiFi* et *Bluetooth* par radio. Cette catégorie de protocoles s'appelle les *protocoles de contrôle d'accès* au support physique, ou protocoles MAC (Medium Access Control). Ces protocoles sont bien sûr plus sophistiqués et plus performant qu'ALOHA. Cependant ils partagent néanmoins le même mécanisme de base qui leur permet de partager efficacement l'accès à un support physique connectant entre eux plusieurs ordinateurs identifiés chacun par une adresse sous forme de suite de bits, et d'envoyer des paquets de données d'un ordinateur à l'autre à travers ce support (câble ou radio).

La différence essentielle entre ALOHA et des protocoles comme Ethernet, Wifi ou Bluetooth est dans la manière de réduire les temps d'attentes avant de retransmettre après une collision, tout en minimisant le nombre et l'impact des collisions, le but étant d'optimiser l'utilisation du support physique lorsque plusieurs ordinateurs ont beaucoup de données à transmettre en même temps.

Dans la suite de ce chapitre, on appellera lien l'abstraction fournie par les protocoles de la couche lien, permettant donc la transmission de paquets de données sur un support physique donné.

Identifiants utilisés à la couche lien

Les identifiants utilisés par la couche lien doivent être cohérents localement, dans le contexte d'un lien. Cette cohérence est assurée si et seulement si chaque identifiant désigne au plus une *interface* connectant un ordinateur à ce lien.

Du point de vue de la couche lien, la destination d'un paquet est l'une de ces interfaces. Le plus souvent, une interface est identifiée par une *adresse MAC*, par exemple dans les cas où la couche lien utilise Ethernet, Wifi ou Bluetooth. Une adresse MAC consiste en une suite de 48 bits, souvent notés de manière plus humaine sous forme hexadécimale, dans un format regroupant des "mots" de 8 bits, comme par exemple 10 :93 :e9 :0a :42 :ac. L'en-tête des paquets envoyés par la couche lien comporte alors l'adresse MAC de l'ordinateur destinataire, ainsi que l'adresse MAC de l'ordinateur émetteur du paquet.

Il existe cependant une exception utile en pratique pour s'adresser à "tout le monde en même temps" : dans certains cas en effet la destination d'un paquet n'est pas une seule interface, mais toutes les interfaces connectées au lien. Dans ce cas, l'adresse de destination indiquée dans le paquet est une adresse MAC spéciale, dite *adresse de diffusion*, désignant par convention "tout le monde connecté à ce lien". Quand un ordinateur reçoit un paquet avec une telle adresse comme destination, il traite le paquet comme si le paquet lui était destiné personnellement.

D'autre part, il existe d'autres formats d'adresses utilisés par des protocoles MAC autres qu'Ethernet, Wifi ou Bluetooth. ATM (Asynchronous Transfer Mode) un protocole MAC utilisé généralement sur fibre optique, met en oeuvre des identifiants de 160 bits. Un autre exemple est Frame Relay, un

autre qui utilise des adresses de longueur variable. Chaque protocole MAC utilise un format d'adresse adapté à la nature du support physique utilisé, au nombre d'ordinateurs maximal pouvant y être connectés, et à la syntaxe utilisée pour former chaque adresse, qui peut fournir dans certain cas des informations supplémentaires comme l'identité de l'industriel qui a construit l'interface réseau etc. Chacune de ces adresses est valide localement et utilisée de manière cohérente sur le lien auquel elle est associée.

On notera finalement que la couche lien n'a pas de résolution d'adresse à accomplir, vu qu'elle est la couche la plus basse utilisant le concept d'adresse. La couche lien est en effet immédiatement supérieure à la couche physique, qui n'a elle-même pas de concept d'adresse, se contentant de transmettre un par un les bits représentant les paquets envoyés par la couche lien, sur le support physique requis.

1.1.5 La couche réseau

On a vu à la section 1.1.2 que les protocoles de cette couche sont responsables de l'aiguillage de chaque paquet vers sa destination, à chaque embranchement entre différents liens rencontrés au cours du périple de ce paquet à travers le réseau. Les protocoles de la couche réseau doivent également identifier les ordinateurs connectés sur le réseau.

Les identifiants utilisés à la couche lien ne sont cohérents que localement, sur le support physique auquel ils sont associés, et leur format varie selon le protocole d'accès utilisé à la couche lien (voir section 1.1.4). Pour identifier les ordinateurs de manière cohérente sur le réseau tout entier, et non plus simplement sur un seul lien, il faut donc utiliser un autre système. Ce système c'est la couche réseau qui le fournit avec le protocole IP (Internet Protocol) qui définit les *adresses IP*, un format d'adresse indépendant des protocoles utilisés à la couche lien, et cohérent à l'échelle du réseau entier. Les adresses IP consistent en 32 bits souvent notés de manière plus humaine sous forme de 4 mots de 8 bits, donc chacun exprimables sous la forme d'un nombre compris entre 0 et $2^8 - 1 = 255$, comme par exemple 216.239.59.104. Ces adresses permettent donc d'identifier un maximum de 2^{32} (soit quatre milliards) ordinateurs de manière unique.

Du point de vue de la couche réseau, la destination d'un paquet est un ordinateur connecté à Internet, identifié donc par une adresse IP, qui figure donc dans l'en-tête des paquets envoyés par la couche réseau. L'en-tête de

ces paquets contient de plus l'adresse IP de l'ordinateur émetteur du paquet. L'en-tête des paquets IP contient de nombreux renseignements en plus des adresses IP, dont notamment deux informations permettant de vérifier la validité du paquet IP. D'une part un code correcteur sous la forme d'une somme de contrôle : un entier codé sur 16 bits, résultat de l'addition des bits constituant le paquet envoyé, qui doit être recalculée et vérifiée comme étant inchangée à l'arrivée du paquet, sinon IP abandonne le traitement de ce paquet car manifestement victime d'une erreur de transmission, qui est donc perdu. D'autre part une durée de vie pour le paquet, un entier codé sur 8 bits, décrémenté d'une unité à chaque fois que le paquet est aiguillé à un embranchement du réseau. Si ce temps de validité devient nul avant que le paquet n'arrive à destination, IP abandonne le traitement de ce paquet car manifestement il ne trouve pas son chemin, et le paquet est donc perdu.

Pour trouver son chemin à travers le réseau de câbles et de liens radio connectant les ordinateurs entre eux, jusqu'à une destination identifiée par son adresse IP, il faut faire appel à un type de protocole supplémentaire, faisant également partie de la couche réseau : un *protocole de routage*. On appelle un ordinateur utilisant un protocole de routage un *routeur* - il existe d'ailleurs au cœur d'Internet des ordinateurs qui sont dédiés au routage, dont les programmes ne font partie que des couches 1, 2 et 3. A contrario, on appelle les autres ordinateurs des *hôtes*, dont les programmes font partie de toutes les couches, dont la couche application utilisée par les internautes. Chaque hôte est connecté à un support physique qui le connecte à au moins un routeur, qui se charge d'acheminer les paquets émis par cet hôte ou destiné à cet hôte, grâce à la connaissance des aiguillages appropriés à l'état actuel du réseau que lui fournit le protocole de routage utilisé. Un hôte, a contrario, n'a pas donc besoin d'avoir cette connaissance, vu que les routeurs s'en chargent et font office de "guichet abstrait" pour utiliser cette connaissance.

Un protocole de routage très simple est le suivant : on fixe l'état du réseau, on observe les différents chemins possible, on en déduit les chemins les meilleurs, et on pré-programme chaque routeur avec une table de règles simplifiées s'apparentant à des panneaux routiers :

- pour aller à E, passer par B, distance= 3
- pour aller à A, passer par B, distance= 2
- pour aller à C passer par D, distance=2
- pour aller à D passer par D, distance=1
- pour aller à B, passer par B, distance= 1

– etc. (une règle par destination possible)

Chaque lettre A, B, C... est en réalité un adresse IP identifiant une destination du point de vue de la couche réseau. Grâce à ce type de table appelée *table de routage*, un routeur peut aiguiller un paquet dans la bonne direction, sachant sa destination finale (une adresse IP), en le relayant sur le lien qui emmène le paquet le plus près possible de sa destination (l'équivalent de prendre le bon embranchement, la bonne route). Cette solution manuelle a cependant un inconvénient majeur : si l'on rajoute ou enlève des ordinateurs ou si on modifie les liens entre eux, il faut reprogrammer tous les ordinateurs. D'autres solutions ont donc été utilisées pour gérer les tables de routage de manière automatisée et plus flexible : des *protocoles de routage*.

Un protocole de routage très simple est le protocole *vecteur de distance*, basé sur l'algorithme de *Bellman-Ford* fonctionnant de la manière suivante. Chaque routeur diffuse périodiquement (disons une fois toute les 30 secondes), sur tous les liens auxquels il est connecté, un paquet spécial appelé *HELLO* contenant sa table de routage actuelle. Vide au départ, cette dernière se remplit au fur et à mesure que le routeur entend les paquets HELLO envoyés par les autres routeurs qu'il entend émettre, et se tient ainsi informé en temps réel de l'ensemble de ses *voisins* : les routeurs avec lesquels il peut communiquer directement via le ou les support(s) physique(s) auxquels il est connecté. Par convention les voisins sont notés dans la table de routage comme étant à distance 1.

De plus, en consultant les tables de routage de ses voisins indiquées dans les HELLO qu'il reçoit périodiquement de chaque voisin, un routeur *entend parler* progressivement d'autres routeurs qui ne sont pas ses voisins, mais des voisins de ses voisins (des routeurs notés à distance 2 dans les tables de routage), puis des voisins des routeurs à distance 2 (donc notés à distance 3 dans les tables de routage) et ainsi de suite, toujours par le truchement de ses voisins directs et de leur HELLO. Il peut ainsi répercuter ces nouvelles informations dans sa propre table de routage en renseignant continuellement via les HELLO qu'il envoie toutes les destinations dont il a connaissance au moment de l'envoi, ainsi que la distance la plus courte pour y arriver dont il a entendu parler jusqu'à présent, et par le truchement de quel voisin.

Ainsi, la table de routage de chaque routeur se construit correctement et puis se tient à jour automatiquement. Le protocole vecteur de distance n'est cependant pas d'une robustesse à toute épreuve, notamment dans certains cas où le protocole dérègle durablement les tables de routage en ne détectant

pas correctement qu'un ou plusieurs routeurs sont soudainement devenu hors-service. Pour cette raison des variantes plus sophistiquées du protocole vecteur de distance ont été développées, et d'autres protocoles utilisant des mécanismes de bases différents ont été inventés. Tous ces protocoles ont en commun de construire et tenir à jour automatiquement une table de routage similaire à celle ci-dessus, et organisent l'aiguillage des paquets selon leur destination suivant les indications contenues dans la table de routage de chaque routeur. Cependant, si la table ne contient pas d'indications concernant la destination d'un paquet à transmettre, le protocole de routage abandonne, et ne traite pas le paquet qui est donc perdu. Un paquet peut également dans certains cas pathologiques "tourner en rond", si les tables de routages sont déréglées (de manière similaire à des panneaux routiers induisant en erreur). Dans ce cas aussi, la couche IP abandonnera le traitement de ce paquet (au bout d'un certain nombre d'aiguillage quand la durée de vie du paquet sera devenue nulle) et le paquet sera perdu.

Pour diminuer la mémoire requise pour stocker les tables de routages, certains routeurs stockent une *route par défaut*, sous la forme d'une règle additionnelle s'apparentant à un panneau routier "toutes directions" indiquant un voisin qui, lui, saura aiguiller vers une destination qui n'est pas explicitement listée. Les plus gros routeurs du réseau, quant à eux, doivent avoir réponse à tout, et n'ont pas de route par défaut dans leur table de routage. Ces routeurs, notamment ceux qui sont au cœur d'Internet, doivent stocker une table de routage qui peut atteindre des centaines de milliers d'entrées, une taille qui n'est pas anodine à gérer. Pour diminuer la taille des tables de routage dans ces routeurs centraux, l'agrégation d'adresses et de préfixes IP est utilisée, comme décrit dans la section suivante.

Préfixes IP

Comme pour les codes postaux, les adresses IP sont organisées de manière hiérarchique. En effet, en comparant deux codes postaux, on peut en général en déduire leur proximité. Par exemple deux adresses postales ayant des codes postaux débutants par les mêmes chiffres seront en général plus proches géographiquement que deux adresses ayant deux codes postaux débutant par des chiffres différents. De manière similaire, la proximité géographique de deux adresses IP est déterminée par leur similarité : plus précisément, deux adresses IP sont d'autant plus proches que la suite de bits qui les débute en commun est longue. Cette suite de bits initiaux en commun est appelée un

préfixe IP. Un préfixe IP est utilisé pour indiquer l'ensemble des adresses IP qui commencent par la suite de bits définie par ce préfixe. Si une adresse IP fait partie de cet ensemble, on dit que l'adresse est issue du préfixe IP. De manière similaire, on peut aussi extraire un préfixe IP d'un préfixe IP de base, le préfixe extrait étant définit par une suite de bit initiaux plus longue que celle définissant le préfixe de base, ce qui correspond à définir un sous-ensemble des adresses appartenant au préfixe de base.

En pratique, un préfixe IP donné est attribué à un lien correspondant, de la manière suivante : tout ordinateur connecté à ce lien est associé à une adresse issue du préfixe IP attribué à ce lien. De manière imbriquée, à l'échelle d'un agrégat de plusieurs liens interconnectés, on associe généralement un préfixe IP à l'agrégat, dont on extrait des préfixes IP plus longs, qu'on attribue chacun à l'un des liens de l'agrégat. On appelle ce procédé l'*agrégation d'adresses IP*.

Grâce à l'agrégation d'adresses IP, il est possible de réduire la taille des tables de routages. Ces dernières peuvent se contenter de lister un préfixe au lieu de lister chaque adresse correspondant à ce préfixe. En effet, vu que toute les adresses issues de ce préfixe sont localisées en gros "au même endroit", elles ont donc en commun la même direction à prendre pour les atteindre.

Résolution d'adresse

Du point de vue de la couche réseau, la destination d'un paquet est un ordinateur connecté à Internet, identifié par une adresse IP, qui figure donc dans l'en-tête des paquets envoyés par la couche réseau. L'en-tête de ces paquets contient de plus l'adresse IP de l'ordinateur émetteur du paquet.

Du point de vue de la couche lien cependant, la destination d'un paquet est un ordinateur connecté directement via un support physique commun, identifié par une adresse MAC utilisant un format différent du format des adresses IP. Vu que l'adresse IP d'un ordinateur peut changer au cours du temps (si l'ordinateur est déménagé ailleurs, par exemple), il n'y a pas de lien fixe entre l'adresse MAC correspondant à une adresse IP donnée.

Pour pouvoir résoudre l'adresse MAC correspondante à une adresse IP, il faut donc utiliser un mécanisme spécifique. Le mécanisme le plus couramment utilisé dans ce but est ARP (Address Resolution Protocol). Son principe est simple : quand un ordinateur demande à découvrir quelle adresse

MAC correspond à une certaine adresse IP, il transmet un paquet spécial sur le lien, signalant au possesseur de l'adresse IP indiquée qu'il doit se manifester par "retour de courrier" indiquant son adresse MAC. Les ordinateurs connectés à ce lien entendent cette transmission et si l'ordinateur utilisant l'adresse IP en question fonctionne normalement, il transmet un paquet réponse indiquant son adresse MAC, à l'intention de l'ordinateur ayant émis la demande. Quand ce dernier reçoit ce paquet réponse, il a résolu l'adresse IP, et il peut noter l'association entre celle-ci et l'adresse MAC correspondante dans une table valide pour un certain laps de temps, pour éviter de devoir redécouvrir systématiquement cette association entre temps (les adresses IP changeant normalement relativement rarement). Au bout de ce laps de temps cette association est effacée, et il faut redemander au possesseur de l'adresse IP en question de se manifester.

Grâce à ARP, la couche réseau peut procéder à la résolution d'adresse, à savoir dans son cas, fournir un identifiant valide à la couche lien, correspondant à l'ordinateur destinataire à savoir une adresse MAC.

Internet, mais sans garantie

Le réseau de câbles et de liens radio connectant les ordinateurs entre eux peut dès à présent être considéré par l'émetteur d'un paquet comme une sorte de câble virtuel le connectant au destinataire du paquet, qui lui permet de communiquer avec la destination de la même manière que si un même câble les connectait directement, même si en réalité, la connexion est indirecte, à travers des ordinateurs et des liens intermédiaires. C'est ce concept de câble virtuel qu'on appelle *Internet*. Dans la suite de ce chapitre, on appellera simplement *réseau* l'abstraction fournie par les protocoles de la couche réseau, permettant donc la transmission de paquets de données à travers une suite de liens.

On notera cepdendant que les transmissions de paquets sur le réseau ne sont pas garanties d'arriver à destination : comme on l'a vu à la section 1.1.4, un paquet peut être perdu en chemin par la couche lien, ou par la couche réseau (voir section).

1.1.6 La couche transport

On a vu dans la section 1.1.2 que cette couche est constituée des protocoles responsables de la paquétisation des données à transmettre, ainsi que de la coordination entre l'envoyeur et le destinataire des paquets, pour assurer la fiabilité de leur transport de bout en bout.

Tout comme il y a un poids maximum autorisé pour une lettre que l'on poste, les paquets de données envoyés sur Internet ont une taille maximale autorisée (par exemple la taille maximale autorisée pour un paquet Ethernet est de 1500 octets, en général). Pour cela, lorsqu'une application requiert l'envoi vers une certaine destination d'un fichier qui ne tient pas en entier dans un seul paquet, il faut le fragmenter en petits bouts qui individuellement, tiennent chacun dans un paquet. On appelle ce procédé la *paquétisation*. Les fragments sont alors envoyés l'un après l'autre vers la destination, chacun dans son paquet individuel, à travers la couche réseau qui peut alors traiter ces paquets l'un après l'autre de manière indépendante. Plusieurs applications peuvent d'ailleurs utiliser en même temps les services de la couche transport, qui doit donc organiser l'équivalent de la levée et de la remise du courrier dans les boîtes aux lettres individuelles qu'elle fournit à chaque application. On appelle une telle boîte au lettre un *port* (noté par un nombre compris entre 0 et 65 535, codé sur 16 bits), et dans ce contexte on appelle *multiplexage* la levée du courrier, et *démultiplexage* la distribution du courrier.

Avec le service postal de base, deux lettres envoyées coup sur coup du même endroit peuvent être reçues dans n'importe quel ordre par leur destinataire. Dans certains cas, une lettre peut même se perdre en route et ne pas arriver du tout. De manière similaire, pour un réseau, une des conséquences notable du procédé "par paquet", est que deux paquets contenant l'information constituant un seul et même fichier à l'origine peuvent arriver dans n'importe quel ordre (donc pas forcément dans l'ordre où ils ont été émis). Certaines fois, un paquet de données peut également se perdre en chemin et ne pas arriver du tout.

Pour pallier ces problèmes potentiels, la couche transport propose des services d'accusés de réception des paquets, et de remise en ordre des paquets reçus conformément à l'ordre dans lequel ils ont été émis. Ces services sont fournis par le protocole *TCP* (Transmission Control Protocol). TCP tient un journal par flux de paquets envoyé vers une même destination, on appelle un tel journal une *connexion TCP*. TCP rajoute un en-tête spécial

à chaque paquet émis dans un tel flux, contenant notamment un numéro de séquence unique dans le contexte de cette connexion TCP. Les numéros de séquence vont croissant d'une unité pour chaque nouveau paquet envoyé vers la destination, qui lorsqu'elle le reçoit, envoie un accusé de réception à l'émetteur mentionnant le numéro de séquence du paquet, et ainsi de suite. Ceci permet de réordonner les paquets à la réception en suivant les numéros de séquences croissants de l'en-tête TCP, au cas où ils ne seraient pas arrivés dans l'ordre. Ceci permet également à l'émetteur de s'assurer que les paquets envoyés sont bien arrivés à destination, en vérifiant qu'un acquittement a bien été reçu pour chaque paquet envoyé, mentionnant le numéro de séquence correspondant à ce paquet. Si un acquittement n'a pas été reçu pour un paquet envoyé, l'émetteur le considère comme perdu et l'envoie de nouveau, en espérant recevoir cette fois un acquittement.

Certaines applications n'ont néanmoins pas besoin de tous les services proposés par TCP. Par exemple, du streaming audio ou video n'a en général pas pas besoin des services d'accusés de réceptions : dans ce contexte, il vaut mieux envoyer de l'image/son actualisé que de renvoyer de l'image/son datant d'il y a quelques secondes. En effet, une certaine perte de données est acceptable dans la mesure où la perception humaine est capable de la compenser, tandis qu'attendre la retransmission de paquets perdus cause des arrêts à répétition qui deviennent vite insupportables.

Pour les applications qui n'ont pas besoin de tous les services proposés par TCP, il existe un protocole alternatif *UDP* (User Datagram Protocol), qui fournit seulement l'équivalent du service postal de base (à savoir le multiplexage et le démultiplexage) aux applications souhaitant utiliser le réseau, en laissant soin à ces applications d'assurer à leur manière tout service supplémentaire.

Identifiants utilisés à la couche transport et résolution d'adresse

Du point de vue de la couche transport, la destination d'un paquet est identifiée par un numéro de port correspondant à l'application destinataire associé à une adresse IP correspondant à l'ordinateur hébergeant cette application. Ces éléments figurent donc dans l'en-tête des paquets envoyés par la couche transport, que ce soit via TCP ou via UDP. L'en-tête de ces paquets contient de plus le numéro de port de l'application émettrice du paquet ainsi que l'adresse IP de l'ordinateur sur laquelle elle s'exécute.

Du point de vue de la couche réseau, la destination d'un paquet est uniquement identifiée par l'adresse IP correspondant à cet ordinateur. La résolution d'adresse est donc très simple à ce stade : il suffit d'extraire l'adresse IP contenue dans l'identifiant utilisé dans l'en-tête de la couche transport.

Contrôle de congestion

On a réussi : Internet est un réseau fiable ! Des programmes sur un ordinateur peuvent ainsi l'utiliser pour envoyer des données binaires à d'autres programmes sur un autre ordinateur, comme si ce dernier et le premier étaient directement connectés par un même câble, même si en réalité la connexion est indirecte à travers plusieurs liens et plusieurs autres ordinateurs. Cependant, pour gérer les variations du nombre d'utilisateurs accédants à Internet en même temps, un mécanisme équivalent à ceux gérant le "temps de parole" vus à la section 1.1.4 est nécessaire, à l'échelle non plus d'un seul câble, mais d'Internet tout entier, qui peut-être considéré à ce stade comme un immense câble virtuel (voir section 1.1.5).

En plus des services vus à la section 1.1.6, le protocole TCP fournit dans ce but un service supplémentaire : l'ajustement du rythme auquel une application envoie un flux de paquets vers une destination donnée. Le but de cet ajustement est de trouver un compromis entre l'intérêt individuel de l'application (envoyer ses paquets le plus rapidement possible pour qu'ils arrivent le plus vite possible), et l'intérêt général de toutes les autres applications utilisant le réseau, à savoir : que ce dernier soit utilisable par tous, et non pas monopolisé par certains. Du point de vue d'une application, Internet est un câble virtuel connectant l'ordinateur sur lequel elle s'execute avec les ordinateurs hébergeant les applications destinataires des données à envoyer. Il lui faut donc découvrir les capacités de transmission vers telle ou telle destination, ces capacités n'étant pas connues à l'avance car elles dépendent de plusieurs paramètres, dont l'état d'engorgement du réseau, qui peut varier de manière extrême d'une seconde à l'autre. La capacité de transmission vers une certaine destination dépend également du chemin utilisé entre émetteur et destination : si pour aller de A à B, par exemple, le câble virtuel est composé d'une suite de supports physiques haute capacité, transportant peu de trafic en ce moment, on pourrait potentiellement envoyer tout de suite beaucoup de trafic de A à B, et ce sans congestion. Si par contre pour aller de A à C on doit passer par une série de liens, dont un lien radio à très faible capacité, on ne pourra envoyer les paquets de A vers C qu'au compte-

goutte. Un autre paramètre dont dépend la capacité de transmission vers une certaine destination est la mémoire actuellement disponible sur chaque ordinateur le long du chemin vers cette destination, dédiée à stocker les paquets qui arrivent souvent plus vite qu'on ne peut les traiter. Cette mémoire est appelée mémoire tampon, et si cette dernière est soudainement saturée par trop de trafic sur l'un des ordinateurs intermédiaire ou sur l'ordinateur destinataire, les paquets suivants sont perdus.

Coordonner l'accès au câble virtuel qu'est Internet de manière centralisée souffrirait en principe des défauts de la rigidité évoquée section 1.1.4 pour l'équivalent à la couche lien. Mais en pratique, ce n'est de toute façon pas envisageable du tout : la gestion en temps réel de centaines de millions d'utilisateurs, bientôt de milliards de machines, et déjà de milliers de milliards de paquets acheminés par seconde, ne peut tout simplement pas être concentrée en un seul point.

Une solution distribuée est donc utilisée par le protocole TCP pour ajuster le rythme auquel une application envoie un flux de paquets vers une destination donnée. Le principe de base est simple, basé sur trois paramètres évalués localement par l'émetteur et le récepteur pour une connexion TCP donnée. Le premier paramètre est W, le nombre de paquets que l'émetteur tolère avoir envoyé vers la destination sans encore avoir reçu d'acquittement les concernant. On appelle ce paramètre la *fenêtre de congestion*. TCP fait varier W dans le temps au gré des détections de pertes de paquets. Le deuxième paramètre est S, le seuil au delà duquel, d'après son expérience, l'émetteur considère que W ne devrait s'aventurer que très prudemment. TCP fait aussi varier S dans le temps au gré des détections de pertes de paquets. Le dernier paramètre est Tw, une estimation du laps de temps au bout duquel on devrait avoir reçu les acquittements correspondant à W paquets envoyés. Un échange d'amorçage entre l'émetteur et la destination permet d'estimer la valeur de Tw, et initialement $W = 1$ et $S = 64$.

L'émetteur répète la procédure suivante, appelée procédure de fenêtre glissante. Dans une première phase, il envoie W paquets qu'on appelle la fenêtre en cours, et attend qu'ils soient tous acquittés. Si cela arrive avant Tw unités de temps, la valeur de W est doublée. L'émetteur envoie alors les W paquets suivants qui deviennent la nouvelle fenêtre en cours et attend leurs acquittements, ainsi de suite jusqu'à ce que $W = S$. Cette phase est appelée *début lent*.

A partir du seuil $W = S$, une deuxième phase s'enclenche, au cours de laquelle à chaque itération, l'émetteur augmente W d'une unité (au lieu

de doubler sa valeur). Cependant, à tout moment au cours la procédure, si l'émetteur doit attendre plus de Tw unités de temps avant d'avoir reçu tous les acquittements de la fenêtre en cours, il considère que des paquets ont été perdu dû à une congestion du réseau. En conséquence, il ajuste S en divisant sa valeur par deux, réinitialise $W = 1$ et recommence depuis le début : doublement de W à chaque itération si la fenêtre en cours est acquittée à temps, jusqu'au seuil $W = S$ à partir duquel W est augmenté d'une unité par itération si la fenêtre en court est acquittée à temps, comme illustré figure 1.7.

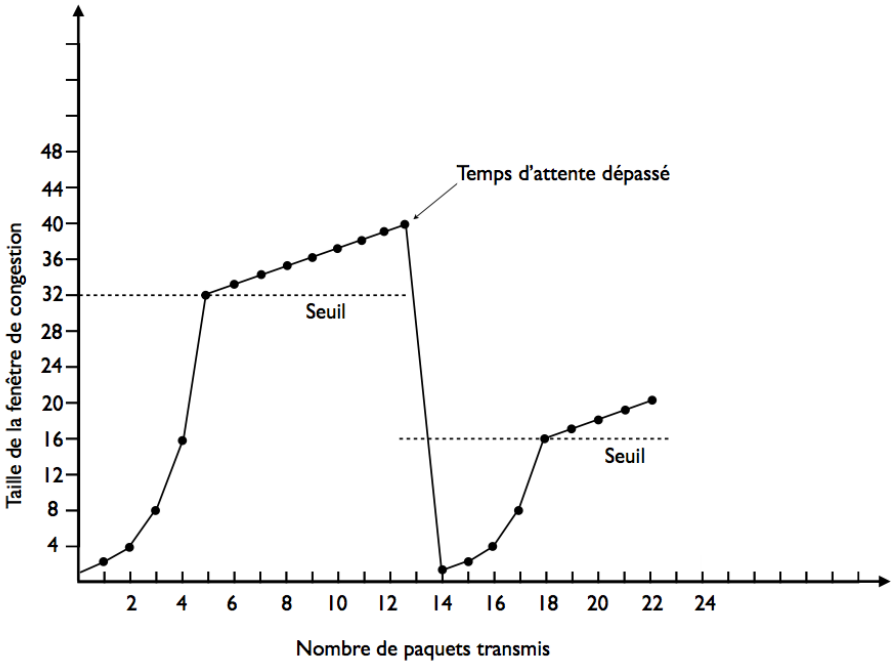


FIGURE 1.7 – TCP. Ajustements progressifs de la cadence d'envois des paquets via les variations de la fenêtre de congestion.

Grâce à ce mécanisme d'ajustements progressifs de la cadence d'envoi des paquets, appelé contrôle de congestion, une application utilisant TCP peut s'adapter en temps réel à ce qu'Internet peut lui fournir comme débit utile instantané pour envoyer un flux de paquets de données. Des versions plus sophistiquées de ce mécanisme sont actuellement utilisées sur Internet, basées sur le même principe, qui permet à un nombre arbitraire d'ordinateurs et d'applications simultanées d'adapter automatiquement leur flux de données

à des débits utiles variant potentiellement de plusieurs ordres de grandeurs. Beaucoup pensent que ce mécanisme, joint au format universel IP, sont les deux principes fondamentaux qui ont permit l'essor d'Internet à l'échelle planétaire, et c'est pourquoi on nomme la pile de protocoles décrite dans ce chapitre comme la pile TCP/IP.

1.1.7 La couche application

Le but de la communication entre ordinateur est de permettre la transmission de données entre applications. Les applications ne sont pas dédiés à faire fonctionner le réseau, et ne sont donc pas abordés en détail dans ce chapitre, si ce n'est ce qui concerne la résolution d'adresse.

Identifiants utilisés à la couche application et résolution d'adresse

Pour s'adapter à leurs utilisateurs humains, la couche application utilise en guise d'identifiants un système hiérarchique de noms tels que *siteweb.com* ou *service.example.fr*, que les gens peuvent mémoriser facilement et utiliser tout les jours pour se connecter à tel ou tel site Internet depuis leur ordinateur, à travers une application telle qu'un navigateur par exemple. Le site internet en question est en réalité une autre application, hébergée sur un autre ordinateur, qui doit donc être contacté via la couche transport, pour pouvoir consulter le site.

Cependant, du point de vue de la couche transport les applications s'exécutant sur un ordinateur distant sont identifiées par un numéro de port d'une part (voir section 1.1.6), et une adresse IP d'autre part (voir section 1.1.5). Les ports identifiant l'application émettrice et l'application destinataire sont connus à l'avance par convention, et peuvent dont être directement renseignés dans l'en-tête des paquets envoyés via la couche transport (voir section 1.1.6). L'application HTTP (HyperText Transfer Protocol), utilisé pour transférer des pages web, a par exemple pour numéro de port attribué le numéro 80. C'est d'ailleurs le système hypertexte constitué des pages reliées par HTTP que l'on appelle le Web, la Toile, ou encore WWW (World Wide Web). Cette dernière dénomination apparait explicitement dans les adresses web de la forme *http ://www.siteweb.com* utilisées par des applications telles les navigateurs.

En revanche, pour permettre plus de fléxibilité, l'adresse IP de l'ordinateur

qui héberge *siteweb.com* n'est pas fixée par convention, et peut changer au cours du temps. Cet ordinateur pourrait par exemple déménager, ou l'application en question pourrait migrer vers un autre ordinateur. Ces cas sont traités de la même manière que pour les adresses postales : si une personne déménage, elle peut ne pas changer de nom, mais il lui faut tout de même changer d'adresse.

Pour pouvoir associer à une addresse web telle *siteweb.com* l'adresse IP de l'ordinateur l'hébergeant, il faut utiliser un mécanisme pour accéder et tenir à jour l'équivalent d'un annuaire, qui associe automatiquement à une addresse web pérenne (par exemple exemple.eu) une adresse IP qui peut donc elle changer au cours du temps comme une personne peut changer de numéro de téléphone. Ce mécanisme est fournit par DNS (Domain Name System), une application qui maintient des copies d'un tel annuaire à différents endroits du réseau connus sous le nom de serveurs DNS, chacun localisé sur un ordinateur identifié par une adresse IP fixée à l'avance, et qui peuvent donc être consultés via Internet par d'autres applications sur d'autres ordinateurs cherchant à résoudre l'adresse d'un nom (exemple.eu dans le cas évoqué ici). L'application DNS a pour numéro de port attribué le numéro 53.

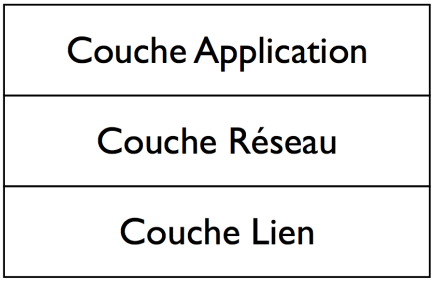
Grâce au DNS et aux numéros de ports connus par convention, la couche application peut procéder à la résolution d'adresse, à savoir dans son cas, fournir un identifiant valide à la couche réseau correspondant à l'application destinataire et à l'ordinateur sur laquelle cette dernière s'exécute, respectivement un numéro de port et une adresse IP.

1.2 Exercices Corrigés

Q1 – En supposant qu'une pile de protocoles soit conçue de telle façon qu'il n'y ait pas de couche de transport dédiée, mais seulement trois couches comme illustré ci-dessous :

Quelle affirmation parmi les suivantes serait correcte (une seule affirmation possible) ?

- ☐ La pile de protocoles requerrait que tous les liens soient 100% fiables (aucune perte et aucune corruption de données).
- ☐ Toutes les machines devraient être connectées par un seul et même support physique, car aucun routage ne serait possible.
- ☐ Seule une application réseau par machine pourrait tourner à la fois.

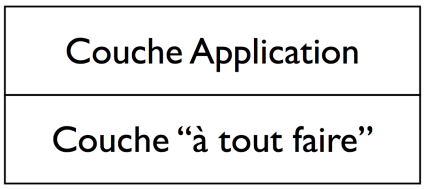


☐ Toutes les affirmations ci-dessus.

☐ Aucune des affirmations ci-dessus.

Correction:
La première affirmation est fausse car il est possible pour chaque application d'inventer son propre mécanisme de transport fiable. La deuxième affirmation est également fausse car établir la connectivité entre machines à travers le réseau est le rôle de la couche réseau, présente dans la pile.
La troisième affirmation est en revanche correcte. Sur Internet, les adresses IP sont utilisées pour identifier un ordinateur destinataire des paquets de données, tandis que les couches transport y ajoutent la notion de port pour identifier l'application destinataire sur cet ordinateur. Sans cette notion et le multiplexage/démultiplexage qui en découle entre la couche réseau et la couche transport, seule une application à la fois pourrait utiliser le réseau, par ordinateur.

Q2 – A chaque fois qu'un paquet passe d'une couche à la couche directement inférieure, des informations supplémentaires sont ajoutées au moyen d'une entête. Ainsi, dans une pile de protocole avec 4 couches, 3 entêtes sont ajoutés (entête transport, entête réseau et entête lien). Supposons maintenant qu'on nous propose la pile de protocoles alternative illustrée ci-dessous, en argumentant qu'elle serait plus efficace car possédant moins de couches et donc ajouterait moins de données supplémentaires à transmettre, tout en fournissant exactement les mêmes services.



Quelle affirmation parmi les suivantes serait correcte (une seule affirmation possible) ?

- ☐ Cette proposition réduit en effet le nombre de données à transmettre.
- ☐ Cette proposition ne réduit pas le nombre de données à transmettre.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Correction:
L'ambition de cette pile de protocoles est de réduire le "coût" supplémentaire introduit par les entêtes ajoutés à chaque couche. Néanmoins, si cette nouvelle couche "à tout faire" doit fournir des fonctionnalités équivalentes, comme par exemple un transport fiable de bout en bout comme fournit par la couche transport, alors la nouvelle couche devrait ajouter dans son entête la même information que celle utilisée par la couche transport . Même argument pour chacune des couches réseau et lien. Au final le nouvelle entête ne serait pas différent d'un entête regroupant transport + réseau + liaison.
Cette pile alternative aurait d'ailleurs des inconvénients : chaque couche fournit un service particulier, via une interface standard bien définie. De cette manière, avec la pile de protocoles TCP/IP, on peut remplacer les mécanismes d'un couche sans avoir à changer les mécanismes des autres couches. Par exemple, remplacer Wifi par Ethernet à la couche lien est possible et très simple : pas besoin de changer quoi que ce soit aux autres couches. A contrario, avec la pile alternative proposée ci-dessus, un tel changement s'avèrerait complexe.

Q3 – Une application utilisant UDP peut-elle bénéficier d'un transfert de données fiable ? (Une seule réponse possible).

- ☐ Non.
- ☐ Oui : mais seulement si les liens entre ordinateurs sont garantis fiables.
- ☐ Oui : mais seulement si TCP est modifié pour s'exécuter au dessus de UDP.
- ☐ Oui : l'application peut fournir son propre mécanisme de transport fiable.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Correction:
UDP est un protocole de transport qui ne garantit pas leur transport fiable de bout en bout : ainsi des paquets peuvent être perdus en transit. Néanmoins, il est tout à fait possible pour une application d'utiliser UDP tout en bénéficiant d'un transport fiable : il suffit simplement que cette application fournisse son propre mécanisme de transport fiable. Ce pourrait être un mécanisme de transport similaire à TCP au moyen de fenêtres glissantes, ou au moyen d'un mécanisme simplifié du même genre, comme par exemple : envoyer un paquet, attendre un accusé de réception avant d'envoyer le paquet suivant, si pas d'accusé retransmettre le paquet.
La quatrième réponse est donc la bonne. La troisième est fausse puisqu'il existe d'autres options de transport fiable que TCP. Enfin, même si tous les liens sont garantis fiables individuellement, un paquet peut quand même être perdu en route. Par exemple, lorsqu'un paquet arrive à un routeur et que celui-ci cesse de fonctionner juste avant de retransmettre le paquet, ce dernier est perdu. De même, si les tables de routages sont temporairement déréglées, un paquet peut suivre un chemin qui ne mène nulle part. La deuxième réponse est donc également incorrecte.

Q4 – En supposant que chaque lien soit 100% fiable (pas de perte ni de corruption de données), quelle affirmation parmi les suivantes est correcte ? (Une seule réponse possible).

- ☐ TCP serait complètement redondant.
- ☐ Le service de livraison fiable des paquets et le mécanisme de contrôle de congestion de TCP seraient redondants.
- ☐ Le service de livraison fiable des paquets de TCP serait redondant, mais pas le mécanisme de contrôle de congestion.
- ☐ Le mécanisme de contrôle de congestion de TCP serait redondant, mais pas le service de livraison fiable des paquets.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

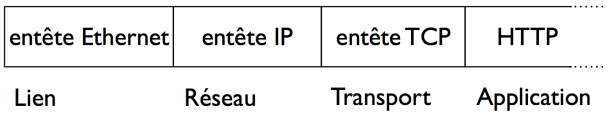
Correction:

Rappelons que TCP fournit plusieurs services, dont : la livraison fiable des paquets, livraison des paquets dans leur ordre d'émission, le contrôle de congestion.
Si tous les liens sur internet étaient fiables, cela n'empêcherait pas deux paquets envoyé coup sur coup de A à B d'arriver dans un ordre différent de l'ordre dans lequel ils ont été envoyé. En effet, rien ne garantit que les deux paquets n'arrivent pas via des chemins différents à travers le réseau, ou que, pour quelque raison que ce soit, un routeur décide de réordonner les paquets avant leur retransmission. Ainsi, la première réponse est fausse, car le service de livraison des paquets dans leur ordre d'émission fourni par TCP n'est pas redondant.
De plus, le fait que les liens soient fiables ne garantit pas qu'un paquet ne soit pas perdu dû à des tables de routages déréglées, ou à un routeur cessant soudainement de fonctionner. Donc le service de livraison fiable des paquets fournit par TCP n'est pas redondant. On en déduit que la deuxième et la troisième réponses sont incorrectes.
Finalement, le fait que les liens soient fiables ne garantit pas que les mémoires tampons disponibles sur les ordinateurs le long du parcours d'un paquet ne soient pas saturées, auquel cas ce paquet ne serait pas traité donc perdu. TCP ne garantit pas non plus que ces mémoires ne soient pas saturées, mais diminue le risque qu'elles le soient grâce au service de contrôle de congestion qu'elle propose. Ce service n'est donc en aucun cas redondant. On en déduit que la quatrième et cinquième réponse sont fausses et que la dernière réponse est correcte
Ainsi, la livraison des paquets dans l'ordre est bien assurée par TCP, qui temporise les paquets reçus jusqu'à pouvoir les délivrer dans le bon ordre à la couche application. Le mécanisme de contrôle de flux de TCP est tout aussi nécessaire car il permet à l'émetteur des paquets de ne pas saturer le tampon de réception du destinataire. Le contrôle de congestion est quant à lui nécessaire pour permettre à l'émetteur d'adapter sa cadence de transmission des données de sorte à ne pas saturer les tampons de réception des routeurs intermédiaires. Enfin, même si tous les liens sont garantis fiables, ce ne sont pas les seuls endroits où les données peuvent être perdues. Par exemple, lorsqu'un paquet arrive sur un routeur et que celui-ci crashe juste avant de retransmettre le paquet, ce dernier est perdu. Ainsi, un mécanisme de livraison fiable des données est tout de même nécessaire. En conclusion, des liens 100% fiables sur internet ne rendent nullement TCP redondant.

Q5 – Un ordinateur est connecté à un lien Ethernet (un câble) et utilise TCP/IP, et l'application HTTP. Quel est le premier octet du contenu d'un paquet envoyé par cet ordinateur sur le câble ? (une seule réponse possible)

- ☐ Le premier octet de l'entête TCP.
- ☐ Le premier octet des données HTTP.
- ☐ Le premier octet de l'entête Ethernet.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Correction:
Dans le contexte décrit par la question, HTTP est un protocole de la couche application, utilisant TCP comme transport. Les paquets TCP sont envoyés via IP du point de vue de la couche couche réseau, et via Ethernet du point de vue de la couche lien (sur le câble). Ainsi le paquet résultant, envoyé par l'ordinateur sur le câble, aura l'allure suivante :



Le premier octet dans le contenu de ce paquet est donc le premier octet du contenu du paquet du point de vue d'Ethernet, c'est à dire le premier octet de l'en-tête IP. Donc toute les réponses sont incorrectes, sauf la dernière.

Q6 – La phase dite de *début lent* de TCP est en fait une période relativement courte, pourquoi ? (une seule réponse possible)

- ☐ Chaque fenêtre de paquets acquittée augmente la taille de fenêtre de congestion, qui croit linéairement et atteint vite le seuil au delà duquel on passe à la phase suivante du mécanisme de contrôle de congestion de TCP.
- ☐ Chaque fenêtre de paquets acquittée augmente la taille de fenêtre de congestion, qui croit exponentiellement atteint vite le seuil au delà duquel on passe à la phase suivante du mécanisme de contrôle de congestion de TCP.
- ☐ Chaque acquittement reçu double la taille de fenêtre, qui croit exponentiellement atteint vite le seuil au delà duquel on passe à la phase suivante du mécanisme de contrôle de congestion de TCP.
- ☐ Aucune des affirmations ci-dessus.

Correction:
La dénomination début lent est presque un abus de terminologie pour désigner la phase initiale de TCP. Certes, pendant cette phase TCP démarre avec une cadence de transmission "lente", mais la cadence augmente exponentiellement, lorsque chaque fenêtre de paquets acquittée double la taille de la fenêtre de congestion, et donc la cadence d'envoi de paquets. Cette croissance exponentielle atteint rapidement le plafond fixé, à partir duquel la croissance est rendue linéaire, plus prudente. La bonne réponse est donc la deuxième réponse. Rappelons qu'à tout moment, si un paquet est perdu TCP interprète cette perte comme un signe de congestion du réseau, et diminue la cadence des envois des paquets : le plafond est divisé par deux et TCP repart à la phase début lent avec une fenêtre de congestion minimale, à savoir égale à un. Ce mécanisme permet de s'adapter automatiquement aux capacités intrinsèques des ordinateurs et liens traversés et en temps réel aux conditions de trafic rencontrées.

Q7 – Imaginons une couche lien au sein de laquelle le protocole MAC se résume simplement à "s'il y a quelque chose à transmettre, transmettre immédiatement même si le support physique est déjà occupé", sans faire aucun effort pour découvrir et réparer les collisions. Imaginons de plus que TCP est le protocole de transport utilisé. Quel serait l'impact de cette couche lien sur les autres couches ? (une seule réponse possible)

- ☐ Aucun impact sur aucune couche excepté sur la couche liaison.
- ☐ Les applications n'auraient d'autre choix que de se préparer à des pertes et corruptions massives de données.
- ☐ Aucun impact sur les applications, car TCP serait en mesure de gérer pertes et retransmissions, et ainsi de de rendre le tout transparent pour les applications.
- ☐ Les applications rencontreraient une baisse considérable des débits car TCP serait obligé d'effectuer de nombreuses retransmissions pour chaque paquet.
- ☐ Les applications rencontreraient une baisse considérable des débits car TCP interpréterait les pertes de données comme une congestion et réduirait la cadence d'envoi des paquets.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Correction:

TCP est capable de gérer les pertes de données en assurant les retransmissions des paquets perdus, donc logiquement aucune couche ne devrait être touchée de manière fonctionnelle. Mais en terme de performance, c'est une toute autre histoire. Rappelons que TCP interprète la perte de données comme un signal que le réseau est congestionné, et en conséquence diminue la cadence d'envoi des paquets : le plafond est divisé par deux et TCP repart à la phase début lent avec une fenêtre de congestion minimale, à savoir égale à un. Dans un réseau où la couche lien ne feraient aucun effort pour découvrir et réparer les collisions, le taux de perte de données serait énorme et par conséquent TCP aurait bien du mal à sortir ne serait-ce que de la phase début lent. Les paquets provenant de l'émetteur risquent même d'entrer en collision avec les accusés réceptions envoyés par le destinataire. Résultat : suite à des pertes de paquets, TCP serait fréquemment forcé de "redémarrer" et par conséquent ne serait jamais amené à atteindre ni de près ni de loin le débit maximum faisable. Ainsi, c'est bien le mécanisme de contrôle de congestion et non pas de retransmission des paquets de TCP qui est à l'origine de la réduction de débit rencontrée par les applications. La cinquième réponse est donc la bonne.

Q8 – Dans un réseau IP (donc sur Internet), un routeur qui n'a pas dans sa table de routage une destination explicite pour un paquet reçu va :

- ☐ Rediriger le paquet vers un routeur par défaut, si une route par défaut est listée dans sa table de routage.
- ☐ Eliminer le paquet, si il n'y a pas de route par défaut listée dans sa table de routage..
- ☐ Mettre le paquet en attente pour retransmission ultérieure.
- ☐ Renvoyer le paquet vers le voisin qui lui a transmis le paquet.
- ☐ Renvoyer le paquet vers l'émetteur du paquet.
- ☐ Aucune des affirmations ci-dessus.

Correction:

Les réseaux IP sont dits "sans garanties", dans le sens où lorsqu'un paquet est prêt à être transmis, il est soit transmis à un ordinateur le rapprochant plus près de sa destination, soit éliminé. La taille des mémoires est un paramètre critique pour les routeurs, et c'est pour cette raison que les paquets ne pouvant être acheminés tout de suite sont éliminés et non stockés. Donc si le routeur n'a pas de route explicite vers la destination d'un paquet, ce paquet est éliminé, ou envoyé vers un voisin listé comme "toutes directions" si un telle route par défaut est en effet présente dans sa table de routage. Donc les deux première réponses sont correctes.
Renvoyer un paquet à rebrousse-chemin (au voisin qui l'a transmis ou à l'ordinateur qui l'a émis au départ) augmenterait la quantité de données échangées à travers le réseau sans bénéfice évident : les ordinateurs en amont du chemin ne disposent probablement pas de chemins alternatifs et ne feront qu'essayer de délivrer le paquet à nouveau par le même chemin, créant ainsi une boucle dans laquelle le même paquet tournerait en rond en passant toujours par les mêmes routeurs sans jamais s'approcher de la destination finale. On en déduit que les trois dernières réponses sont fausses.

1.3 Exercices

Les exercices suivant utilise l'algorithme de routage suivant, basé sur l'algorithme de Bellman-Ford :

Définitions :
- Soit u le routeur exécutant l'algorithme, et V l'ensemble des routeurs dans le réseau.
- Soit $distance(w, v)$ la distance la plus courte (en nombre de liens intermédiaires), connue jusqu'à présent, de u à w via un voisin v de u.
- Soit $distance(w, *)$ la distance la plus courte (en nombre de liens intermédiaires), connue jusqu'à présent, de u à w, via n'importe quel voisin de u.
- Soit $c(u, v)$ telle que $c(u, v) = 1$ si un lien direct existe entre u et v, et $c(u, v) = \infty$ si aucun lien direct n'existe entre u et v.

Initialisation :

1. $\forall (v, w) \in V : distance(w, v) = \infty$
   C'est à dire, le routeur u ne connaît rien du réseau initialement.
2. $\forall v \in V | c(u, v) \neq \infty : distance(v, v) = c(u, v)$
   Le routeur u enregistre la distance entre lui et ses voisins, c'est à dire les routeurs avec lesquels il a un lien direct (via un câble par exemple), et note intuitivement le fait que, pour atteindre un voisin, il faut utiliser comme prochaine étape le voisin en question.
3. $\forall w \in V \setminus \{u\} | distance(w, *) \neq \infty$: envoyer $(w, distance(w, *))$ à chaque voisin
   Le routeur u détermine la plus courte distance qu'il connaît entre lui et chaque destination dans le réseau. L'ensemble des paires (destination, distance la plus courte vers cette destination) ainsi calculées par le routeur constitue le *vecteur de distance* de ce routeur. Le routeur u envoie son vecteur de distance à chacun de ses voisins. Ces derniers connaissent désormais la distance entre u et toutes les destinations du réseau.

Répéter à l'infini :

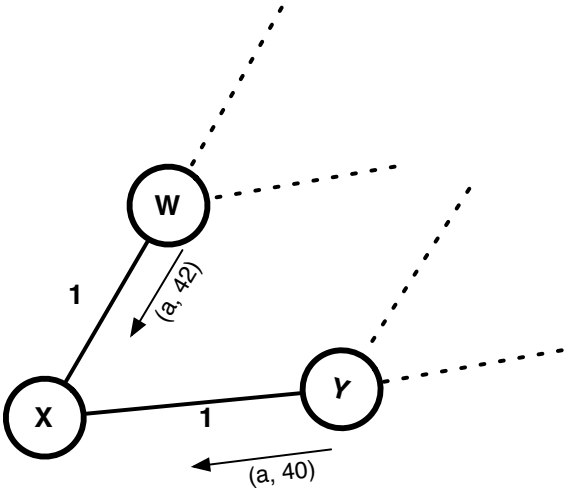
1. Attendre l'un des événements suivants :
   - réception d'un vecteur de distance émis par un des voisins de u.
   - détection d'un changement de coût d'un lien avec voisin v.
2. Si le coût d'un lien entre u et v change de $c(u, v)$ à $c'(u, v)$ (*par exemple le coût devient ∞ quand un lien disparaît*), alors :

- $\forall w' \in V : distance(w', v) = distance(w', v) + (c'(u, v) - c(u, v))$
  Toutes les distances calculées via v, sont mises à jour pour refléter le nouveau coût du lien $c'(u, v)$.

3. Si un vecteur de distance est reçu d'un routeur voisin v, déclarant $(w, distance(w, *))$, alors :
   - $distance(w, v) = distance(v, *) + distance(w, v)$
4. Si n'importe quelle distance $distance(w, *)$ a changé :
   - $\forall w \in V \setminus \{u\} | distance(w, *) \neq \infty$: envoyer $(w, distance(w, *))$ à tous les voisins de u

   Si le routeur détermine que son vecteur de distance a changé, alors il renvoie son nouveau vecteur de distance à chacun de ses voisins.

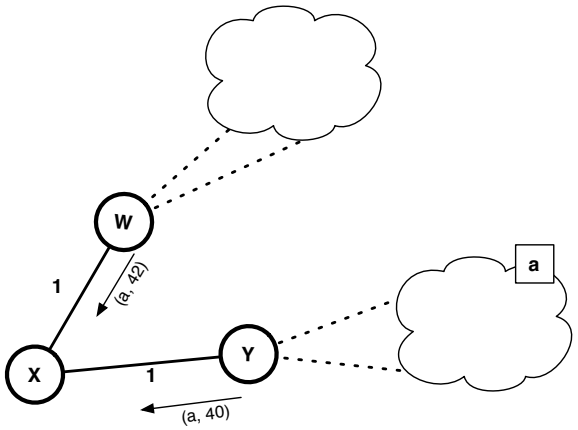
Q9 – Considérons le réseau représenté sur la figure ci-dessous. Seuls 3 routeurs sont représentés et les pointillés indiquent qu'ils sont reliés à un réseau plus grand. Les liens entre les routeurs X, W et Y sont indiqués par des traits pleins. Le routeur X reçoit un message avec le vecteur de distance de Y indiquant (a,40), et un message de W indiquant (a, 42). Il n'y a pas d'autres liens partant de X que ceux indiqués sur la figure. Quelle affirmation parmi les suivantes est correcte pour le réseau ainsi représenté ? (une seule réponse correcte)



- ☐ Le routeur X va enregistrer Y comme le prochaine étape sur le chemin le plus court vers a, avec un coût de 40.
- ☐ Le routeur X va enregistrer Y comme le prochaine étape sur le chemin le plus court vers a, avec un coût de 41.
- ☐ Le routeur W a enregistré X comme le prochaine étape sur le chemin le plus court vers a, avec un coût de 42
- ☐ Le routeur W a enregistré a comme le prochaine étape sur le chemin le plus court vers Y, avec un coût de 43
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Q10 – Considérons le réseau de la figure ci-dessous. Seuls 3 routeurs sont représentés et les pointillés indiquent que les routeurs W et Y sont chacun

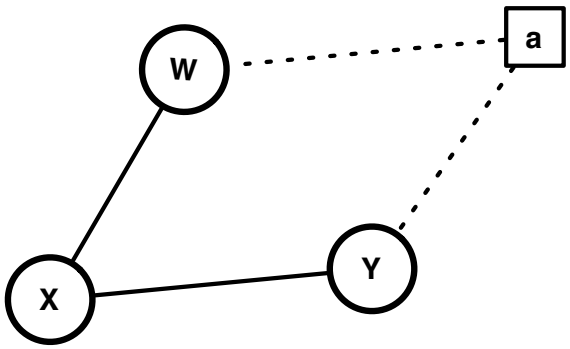
connectés à un réseau distinct plus grand. Les liens entre les routeurs X,W et Y sont indiqués par des traits pleins. De plus, le routeur X reçoit un vecteur de distance de Y indiquant $(a, 40)$ et un de W indiquant $(a, 42)$. Notons que la destination a est atteignable uniquement via Y. Supposons maintenant que le lien entre le routeur Y et le routeur X est soudainement rompu, de manière permanente, c'est à dire $c(X, Y) = \infty$. Quelle affirmation est correcte parmi les suivantes ? (une seule réponse possible)



- ☐ Le routeur X va découvrir que le coût de tous les chemins via le routeur Y est maintenant ∞, et par conséquent ignorer tous les chemins passant par Y.
- ☐ Le routeur X va immédiatement découvrir que a n'est plus atteignable et le fait savoir à W en lui envoyant un message indiquant que la distance à a via X est ∞.
- ☐ Le routeur X va immédiatement découvrir que a n'est plus atteignable et le fait savoir à W en lui envoyant un message indiquant que la distance à a via X est ∞. Le routeur W propage ensuite immédiatement cette information aux autres liens auxquels il est connecté.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Q11 – La version simplifiée du protocole vecteur de distance vue jusqu'ici suppose initialement une distance $c(u, v) = 1$ si un lien direct existe entre

u et v, et sinon $c(u, v) = \infty$. Une version plus élaborée pourrait prendre en compte des valeurs initiales plus variées, à savoir : $c(u, v) \in [1; \infty[$ si un lien direct existe entre u et v, et sinon $c(u, v) = \infty$. Une version encore plus élaborée pourrait prendre en compte des variations de coût par lien, $c(u, v)$ qui pourrait par exemple augmenter proportionnellement en fonction de la quantité de trafic qu'il véhicule (dans le but d'encourager l'équilibre de la charge de chaque lien dans le réseau). Dans le réseau décrit ci-dessous, le but d'une telle élaboration serait d'envoyer le trafic de X vers a à la fois via $X - W - a$ et via $X - Y - a$ en parallèle, pour maximiser le débit du flux entre X et a.



Dans ce cas, quelle affirmation est correcte parmi les suivantes ? (une seule réponse possible)

- ☐ Les tables de routages pourraient ne pas jamais converger, à savoir atteindre un état stable et cohérent entre elles.
- ☐ Le trafic n'utilisera pas les deux chemins en parallèle, mais l'un ou l'autre, en tout cas pas les deux.
- ☐ Le trafic n'utilisera pas véritablement les deux chemins en parallèle, mais alternera indéfiniment entre les deux.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

Q12 – Un exemple d'application simple est TELNET (TErminaL NETwork), qui offre un moyen de communication bi-directionnel généraliste entre deux machines distantes. Ouvrir un terminal, et exécuter l'application *TELNET* via la ligne de commande, en tapant *telnet www.google.com 80*, puis taper sur la touche ENTER. Il s'affiche le texte suivant :

```
Trying 74.125.230.80...
Connected to www.l.google.com.
Escape character is '^]'.
```

Quelle affirmation est correcte parmi les suivantes ? (une seule réponse possible)

- ☐ *TELNET* est une application qui accède au réseau via la couche transport sur l'ordinateur.
- ☐ L'addresse IP 74.125.230.80 est l'adresse qui correspond au site www.google.com, trouvée grâce à DNS.
- ☐ Dans la commande *telnet www.google.com 80*, le nombre 80 est le numéro de port visé sur l'ordinateur destinataire.
- ☐ Le protocole de transport utilisé par TELNET est TCP.
- ☐ Toutes les affirmations ci-dessus.
- ☐ Aucune des affirmations ci-dessus.

1.4 Questions d'enseignement

L'Internet d'aujourd'hui, basé essentiellement sur IPv4, permet d'identifier et de faire communiquer ensemble 2^{32} machines différentes, soit un peu plus de 4 milliards d'ordinateurs (routeurs et hôtes). Un autre réseau de taille à peu près comparable est le cerveau humain, qui connecte quelques 100 milliards de neurones en moyenne. Ce dernier a évolué depuis des millions d'années avant d'arriver à sa taille et à sa complexité actuelle, une éternité comparée aux 40 ans seulement depuis les prémices d'Internet !

Internet est déjà, sans doute, la plus complexe et la plus vaste construction jamais réalisée par l'homme. Depuis 2011, il n'y a plus d'adresse IPv4 qui ne soit pas allouée : Internet a atteint la taille adulte, et change actuellement de garde-robe, grâce à des protocoles et des formats rénovés (IPv6), qui visent à pouvoir faire communiquer ensemble jusqu'à 2^{128} (soit 3.10^{38}) machines, soit plusieurs ordres de grandeur de plus que le nombre d'Avogadro !

Ce chapitre tente d'introduire les notions essentielles, formant la base du système immensément complexe qu'est Internet. Pour eveiller l'intérêt, comparer la complexité de ce système avec celui du cerveau d'un être humain

est peut-être plus profondément motivant encore que d'interpeller à brûle-pourpoint les élèves sur le dernier sujet à la mode, comme par exemple : "Vous êtes vous demandé comment marche Facebook réellement ?".

1.4.1 Gérer la Complexité au Moyen d'Abstractions

Quand on écrit un courriel à `personne@exemple.com`, on ne sait pas où cette personne est physiquement, elle pourrait tout aussi bien être de l'autre côté de la terre, ou dans la pièce d'à côté. Même l'ordinateur depuis lequel on écrit ce courriel ne sait pas où se trouve la machine stockant les courriels de `personne@exemple.com`. L'important est que l'ordinateur n'a *pas besoin de le savoir*. L'ordinateur n'a pas non plus besoin de savoir si la machine stockant les courriels de `personne@exemple.com` est joignable via Lyon ou via Berlin.

Une manière de gérer la complexité du système est d'avoir une séparation claire entre deux catégories de machines sur Internet : les hôtes d'une part, et les routeurs d'autre part, la périphérie du réseau d'une part, et le coeur du réseau d'autre part, comme illustré dans la figure 1.8.

Les hôtes sont des machines telles que les ordinateurs utilisés par un internaute de tous les jours, les smartphones, mais aussi les serveurs mail, les serveurs web etc. Leur rôle est essentiellement *d'utiliser Internet*, en exécutant des applications accédant au réseau via la couche transport. Du point de vue réseau, un programmeur d'application se soucie essentiellement de savoir quel protocole de transport utiliser (TCP ? UDP ?), et quels sont les noms des hôtes ou serveurs avec lesquels communiquer. Le programmeur ne se soucie pas du reste, notamment pas de comment les paquets trouvent leur chemin, jusqu'à ces hôtes ou serveurs, et s'en remet au réseau pour cela.

Et le réseau s'en charge ! Quand on configure un ordinateur pour accéder à Internet, on renseigne essentiellement ce dernier de l'adresse du "routeur par défaut" à contacter pour tout accès réseau. Ce routeur est justement à la frontière entre la périphérie et le coeur d'Internet, et la configuration de l'ordinateur concernant l'utilisation de ce routeur par défaut est agnostique quant à l'application ou le protocole de transport utilisés pour communiquer par l'intermédiaire de ce routeur avec un serveur ou un hôte distant, que cet hôte ou serveur soit dans la pièce d'à côté ou à l'autre bout du monde. De la même manière que pour la communication entre humains (voir la transcrip-

tion d'interview illustrée figure 1.2) l'intermédiaire est agnostique quant aux idées échangées, et se contente de retranscrire l'oral en écrit, dans les réseaux d'ordinateurs, les routeurs se contentent d'aiguiller du mieux possible, les paquets que la couche transport lui demande d'envoyer vers leur destination. Pour une application sur un ordinateur, elles reçoivent et envoient des données, et tout ce qu'il se passe au delà du routeur par défaut, jusqu'à destination, c'est égal : les routeurs s'en chargent, que ce soit via Ethernet, Wifi, voire des signaux de fumées et/ou des pigeons voyageurs.

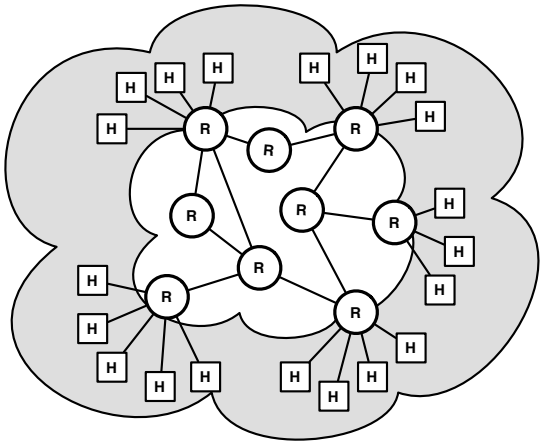


FIGURE 1.8 – Coeur et périphérie. Routeurs et hôtes.

1.4.2 Comment Enseigner

Il y a essentiellement deux types d'approche didactique pour introduire le domaine des réseaux d'ordinateurs en général et Internet en particulier.

Construire de bas en haut

La première approche didactique est de construire le système à partir de rien pour arriver à une système fonctionnel. Le déroulé typique de cette approche est de commencer par décrire comment faire le premier pas, à savoir envoyer des informations sur un lien (câble ou radio). Puis on décrit le deuxième pas : le réseau de liens. Comment l'information trouve son chemin à travers un dédale de liens, et comment l'information est transférée automatiquement le

long de ce chemin, même si les liens qui composent ce dernier sont de natures diverses ? On passe ensuite aux protocoles de transports à l'oeuvre au dessus du réseau et les services de gestion et de garanties qu'ils fournissent. Enfin on aborde les applications qui utilisent les services de la couche transport. Cette approche didactique se résume essentiellement à la démarche suivante : à partir d'arêtes (les liens) et de sommets (les routeurs) on construit un graphe, sur lequel des algorithmes sont mis à l'oeuvre.

La plupart de la littérature adopte cette approche, qui a l'avantage d'avancer peu à peu sur des acquis clairs et solides, ce qui n'est pas négligeable. Cependant, cette approche comporte certains risques, dont il faut être conscient en tant qu'enseignant :

Un long chemin avant de pouvoir pratiquer - pour pouvoir comprendre comment programmer une application simple utilisant le réseau, il faut avoir absolument tout étudié : les couches lien, réseau, transport. Les détails des couches basses ne sont pas forcément primordiaux pour les non-spécialistes, tandis que comprendre les couches supérieures est une priorité. En effet la plupart des dernières évolutions d'Internet se sont notamment passées à la couche application, que ce soit le web, Skype, les réseaux sociaux, les réseaux pair-à-pair, la video à la demande etc.

L'envers du décor avant les concepts clés - dans un certain sens, enseigner les réseaux d'ordinateurs en construisant de bas en haut est contraire à ce que cette construction même cherche à produire, à savoir : une architecture d'abstractions permettant de s'affranchir de certaines complexités. En construisant de bas en haut, les élèves sont exposés au complexités d'abord, et seulement ensuite aux abstractions architecturales, alors que ces dernières sont fondamentales.

Conserver l'intérêt des élèves peut être ardu - ne pas tomber dans un catalogue assommant (pour les élèves comme pour le professeur) de formats et propriétés peut s'avérer difficile. On perd facilement la vue d'ensemble qu'il est nécessaire de garder à l'esprit pour comprendre l'utilité des algorithmes et des protocoles dont on parle.

Déconstruire de haut en bas

La deuxième approche didactique est moins scolaire : il s'agit de déconstruire le système à partir de la couche application. Le déroulé typique de cette approche est de commencer par l'interaction client-serveur pour HTTP ou pour

le courriel, et donner l'exemple de l'application Telnet qu'on peut manipuler en travaux pratiques. Ensuite, toujours en travaux pratique, on peut faire écrire aux élèves un programme dans un langage qu'ils connaissent, similaire du point de vue fonctionnalité à Telnet, envoyant et recevant des données via TCP, fournissant par exemple de la messagerie instantanée entre deux ordinateurs. On peut alors aborder les services de la couche transport, par analogie avec les services postaux (service basique, recommandé avec accusé de réception, Chronopost), et le rôle de la couche réseau, puis le concept même d'organisation en pile de protocoles. Si les élèves se prennent au jeu, une séance interactive peut venir assez naturellement, au cours de laquelle on peut demander aux élèves de suggérer eux-mêmes des solutions ou des améliorations aux différente solutions pour les services de la couche transport. La suite se concentre sur les différentes solutions possibles pour fournir les services de la couche réseau, introduire la différence entre hôtes et routeurs etc. ce qui peut donner lieu à une autre séance interactive sur le sujet. On peut alors conclure sur le rôle et les mécanismes de la couche lien.

Les avantages principaux de cette approche sont de pouvoir s'appuyer sur des travaux pratiques motivants dès le début du cours (pourvu que des ordinateurs connectés à Internet soient à disposition), et d'avoir l'occasion de faire comprendre aux élèves l'abstraction d'abord, et de découvrir les détails sous-jacents à l'abstraction ensuite. Cependant, cette approche comporte certains risques, dont il faut être conscient en tant qu'enseignant :

Déconstruire peut être frustrant - il faut attendre la fin du cours pour avoir des explications sur certains mécanisme de base d'Internet, ce qui peut s'avérer inconfortable pour ceux qui cherchent d'emblée le germe de la "réaction en chaîne" qui produit Internet.

Le côté non-scolaire peut dérouter - séances interactives et travaux pratiques sont d'excellents catalyseurs, mais ne sont pas forcément à la portée de tous dans toutes les circonstances, du point de vue des élèves comme du point de vue du professeur.

1.4.3 Les Points Clés

L'objectif pour les élèves est d'avoir acquis les notions de bases concernant l'architecture matérielle et logicielle d'Internet. Il est important d'avoir compris comment une pile de protocole fonctionne, et le rôle de chaque couche. Il est également important d'avoir compris le rôle des protocoles de transport sur les hôtes en périphérie d'Internet, et le rôle des routeurs au coeur

d'Internet, qui gèrent l'aspect algorithmique sur le graphe sous-jacent du réseau.

Pour finir, il est important que les élèves acquièrent le sentiment d'avoir appris quelque chose en prise avec le réel, par exemple par le biais de la programmation d'une application simple communicant sur le réseau. De ce sentiment pourrait germer chez les élèves l'idée qu'ils ne sont pas seulement consommateurs, mais bien des acteurs potentiels d'Internet – ce qui est la réalité.

1.4.4 Compléments

L'informatique en général, et les réseaux informatiques en particulier, utilisent en pratique des termes anglais et des acronymes provenant d'expressions anglaise. Il est donc impératif d'apprendre les termes anglais en même temps que les termes français, afin de pouvoir directement appliquer les savoirs acquis, ne serait-ce que pour bien comprendre ou modifier la configuration de l'accès réseau d'un ordinateur personnel. Ci-dessous un bref lexique, à compléter selon les besoins :

Byte : Octet.

Congestion Window : Fenêtre de congestion (de TCP).

Data : Données.

Header : Entête.

Host : Hôte.

Layer : Couche.

Link : Lien.

Medium : Support physique.

Network : Réseau.

Packet : Paquet.

Prefix : Préfixe.

Protocol : Protocole.

Router : Routeur.

Slow Start : Début lent (de TCP).

Stack : Pile.

... etc.

De même, la plupart des spécification des normes et des protocoles sont disponibles uniquement en anglais. On en conclut donc qu'une certaine maîtrise de la langue anglaise est nécéssaire en pratique.

Normes de la Couche Lien

Les protocoles Ethernet, Wifi ou Bluetooth mentionné 1.1.4, sont basés sur une évolution des mécanismes de ALOHA, appelée CSMA (Carrier Sense Multiple Access), qui permet à un ordinateur d'éviter plus de collisions, en l'obligeant à écouter le support physique avant de transmettre pour vérifier qu'aucun autre ordinateur n'est déjà en train de transmettre, dans quel cas il sursoit, et attend un temps aléatoire avant de recommencer la procédure (écouter avant d'essayer de transmettre). Une variante de CSMA appelée CSMA-CD (Collision Detection) est à la base d'Ethernet, et réduit l'impact des collisions en obligeant un ordinateur à écouter pendant qu'il transmet, pour détecter si une collision est en train d'avoir lieu et dans ce cas arrêter immédiatement de transmettre (au lieu de finir de transmettre le paquet prévu, qui de toute façon est perdu dû à la collision). Une autre variante, CSMA-CA (Collision Avoidance), est à la base de Wifi, et permet d'esquiver plus de collisions.

Ethernet, Wifi et Bluetooth sont des normes de communications locales entre machines, spécifiées par un organisme appelé l'IEEE (acronyme anglais pour Institute of Electrical and Electronics Engineers). Le principe d'une norme est de publier un modus operandi garantissant la compatibilité avec certains critères, pourvu que ce modus operandi soit mis en œuvre. Une norme de communication entre machines, par exemple, consiste en une spécification consultable publiquement, pouvant ainsi être utilisée par n'importe quel constructeur d'ordinateur pour garantir que ses ordinateurs peuvent communiquer avec n'importe quel autre ordinateur, même issu d'un autre constructeur, pourvu que la spécification soit aussi respectée par cet autre constructeur. Pour consulter les spécifications d'Ethernet, Wifi ou Bluetooth, il suffit de connaître le nom technique de chacune de ces normes, à savoir IEEE 802.3 (Ethernet), IEEE 802.11 (Wifi), et IEEE 802.15.1 (Bluetooth). D'autres organismes de normalisation existent, tels 3GPP, qui s'occupe des normes de téléphonie cellulaire, qui utilisent des mécanismes différents pour connecter des appareils au réseau téléphonique et à Internet, tel UMTS (nom technique de la 3G).

ARPANET : la Naissance de la Couche Réseau

L'aiguillage des paquets de manière indépendante les uns des autres, pouvant prendre alternativement des chemins différents pour aller au même endroit, est appelé la commutation par paquet. Il existe une autre technique pour aiguiller les paquets, basée sur l'établissement d'un chemin physique ou logique fixé entre deux ordinateurs, le temps d'une session de communication entre ces deux ordinateurs, le long duquel sont transférés tous les paquets appartenant à cette session. Cette technique s'appelle la commutation de circuits, et est issue des réseaux téléphoniques, où le concept de session (l'occupation d'une ligne téléphonique le temps d'une conversation) a un rôle prépondérant. Le défaut majeur de la commutation de circuit est qu'il n'est pas possible d'utiliser les "silences" sur la ligne d'une conversation pour faire passer d'autre paquets d'autres conversations, ce qui sous-utilise le réseau, contrairement à la commutation de paquets, qui peut mieux utiliser les capacités du réseau. C'est pourquoi la commutation de circuit est peu à peu délaissée.

La commutation de paquets a été utilisée pour la première fois à la fin des années 1960 dans un réseau appelé ARPANET, l'ancêtre d'Internet, développé par les États-Unis pendant la Guerre Froide afin d'unifier les techniques de connexion permettant à un terminal de communiquer à distance avec des ordinateurs de constructeurs différents. Selon un mythe répandu à propos d'ARPANET, ce dernier aurait été projeté pour doter le pays d'un réseau plus résistant que le réseau téléphonique, le but étant de pouvoir continuer à fonctionner quel que soit l'état de destruction du pays, même suite à une attaque nucléaire.

L'archétype de la commutation par paquet est Internet, basé sur le protocole IP. Ce protocole est une norme de communication entre machine, spécifié par l'organisme de normalisation appelé IETF (acronyme anglais de Internet Engineering Task Force). Il existe deux versions du protocole utilisé actuellement. L'une d'entre elles est IPv4, la version du protocole IP présentée dans ce chapitre. IPv4 est accompagnée d'une suite d'autres protocoles avec lesquels il fonctionne de concert pour former Internet tel que la plupart des internautes d'aujourd'hui le connaissent. Il existe néanmoins une autre version, appelée IPv6, qui est une version rénovée du protocole IP. IPv6 est également accompagné de sa propre suite de protocoles avec lesquels il fonctionne de concert pour former une partie d'Internet moins connue des internautes, mais néanmoins importante car plus à même de

gérer la taille gigantesque qu'Internet a acquise, et sa croissance encore à venir. En effet, le nombre de machine connectées à Internet étant devenu énorme, les 2^{32} adresses IPv4 disponibles n'étaient plus suffisantes. IPv6 a donc été développé en utilisant des adresses de 128 bits, permettant d'identifier beaucoup plus de machines (au maximum 2^{128}). Cependant, la transition entre l'utilisation d'IPv4 et IPv6 n'est pas simple à opérer à grande échelle, et suscite de nombreuses questions depuis quelques années.

La suite des protocoles IPv4, ainsi que la suite des protocoles IPv6 sont publiés dans des documents accessibles publiquement sur Internet, dans une série de documents appelés RFC (acronyme anglais pour Request For Comments). IPv6, par exemple est spécifié dans la RFC 246, tandis que IPv4 est spécifié quant à lui dans la RFC 791. La figure 1.9 illustre la structure de l'en-tête d'un paquet IPv4. On y retrouve les champs correspondants aux adresses IP de l'émetteur du paquet (la source) et de la destination du paquet, ainsi que la somme de contrôle et la durée de vie du paquet IP, que l'on a vu dans cette section. Ces champs ainsi que tous les autres, sont définis dans la RFC 791.

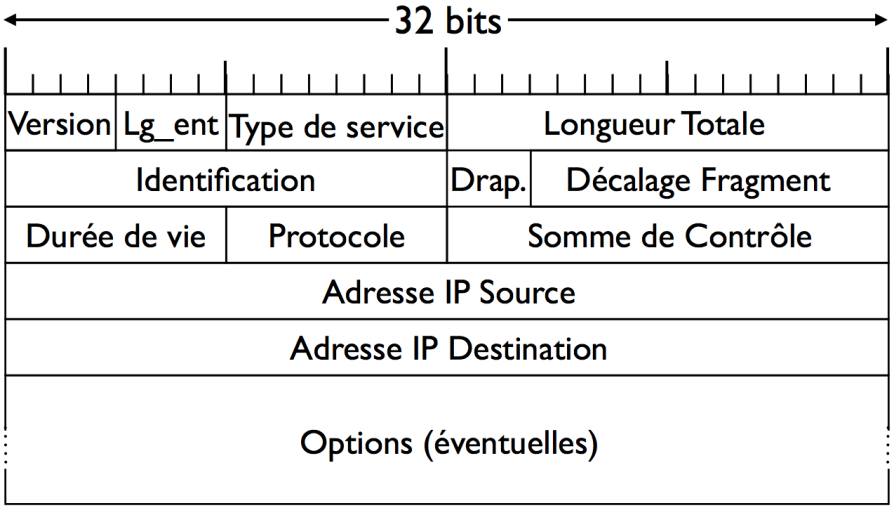


FIGURE 1.9 – Structure de l'en-tête d'un paquet IPv4, illustrées ici par rangées successives de 32 bits lues de gauche à droite. L'adresse source et l'adresse destination sont les adresses IP de l'émetteur et du destinataire du paquet, respectivement.