Prométhée Spathis

Technologies et protocoles Internet

Avant-propos de Jim Kurose

ellipses

Chapitre 1

Architecture des réseaux

Pour comprendre le fonctionnement d'un système, on doit comprendre sa raison d'être. Dans le cas des réseaux informatiques, il s'agit de transporter des données. À l'origine de ces données sont les applications que les utilisateurs exécutent sur leur smartphone, tablette ou autres appareils connectés.

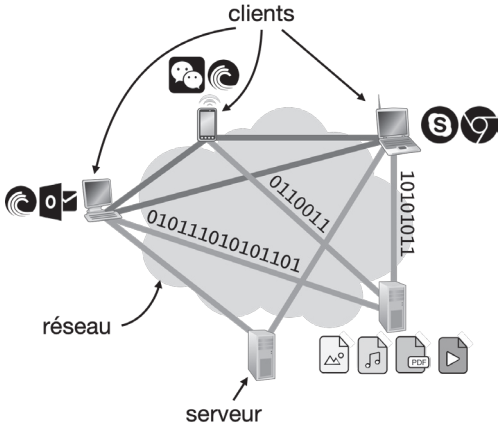

Figure 1.1. Applications et machines hôtes. Les machines hôtes situées à la périphérie du réseau hébergent les applications. Dans le cadre des applications client-serveur, on distingue les serveurs qui proposent des contenus et des services que les clients obtiennent en soumettant des requêtes.

1.1 | Applications Internet

Les données font référence aux bits que génèrent les applications exécutées à l'initiative des utilisateurs sur leur ordinateur. Des exemples d'applications sont les navigateurs Web, les clients mail ou les clients BitTorrent. Toutes ces applications sont des programmes qui communiquent avec un programme homologue, exécuté sur une machine distante.

La Figure 1.1 montre l'exemple d'un réseau où des machines d'extrémité exécutent des applications, pour le compte desquelles le réseau véhiculent des données.

1.1.1 Modèle client-serveur

Originellement, les applications étaient conçues de manière asymétrique : le programme distant diffère de celui exécuté sur l'ordinateur des utilisateurs. En tant qu'utilisateurs, nous nous connectons à un réseau de données en vue de recevoir une page Web, des images, un email ou une vidéo.

Ces objets sont hébergés par des machines dont nous sommes les clients et qui, du fait de répondre à nos requêtes, sont appelées serveurs.

Les serveurs sont dotés de capacités de stockage qui leur permettent d'héberger des contenus accessibles de tous. Les serveurs disposent également de capacités de traitement qui leur permettent de traiter les requêtes des clients. Les applications conçues selon ce modèle, sont appelées applications client-serveur.

1.1.2 Modèle pair-à-pair

Récemment, on a vu l'avènement d'un nouveau type d'applications résultant de l'exécution de programmes identiques. On parle d'applications pair-à-pair (*peer-to-peer* ou P2P). Conformément à ce modèle, une même machine peut agir aussi bien en tant que client que serveur.

Les applications de partage de fichiers telles que Napster ou BitTorrent sont des exemples d'application conçues selon un modèle pair-à-pair.

1.1.3 Interopérabilité

Applications client-serveur ou pair-à-pair sont mises à disposition au téléchargement par leurs développeurs. Les utilisateurs sont libres d'installer ces applications sur leur ordinateur. Il n'y a pas, à proprement parler, de réel contrôle concernant les applications qu'un utilisateur peut installer sur son ordinateur.

Qu'ils s'agissent des constructeurs de matériel informatique ou des concepteurs des systèmes d'exploitation qui équipent nos ordinateurs à leur achat, tous œuvrent dans le but de concevoir des équipements ouverts et interopérables qui rendent leurs produits attractifs du fait de leur versatilité. C'est ce qui a fait le succès de l'Internet.

Pour assurer l'interopérabilité des applications, certaines ont été standardisées permettant ainsi à plusieurs sociétés d'édition de logiciels de développer des applications concurrentes. Du fait d'être développés conformément aux standards, ces applications sont interopérables.

L'utilisation de certains logiciels peut être restreinte. Un fournisseur d'accès Internet (FAI) peut éventuellement être soumis aux lois nationales contre le piratage informatique ou la violation des droits d'auteur, par exemple. Pour autant, les FAI ne sont pas toujours en mesure de filtrer avec succès, les trafics dérogeant à ces lois.

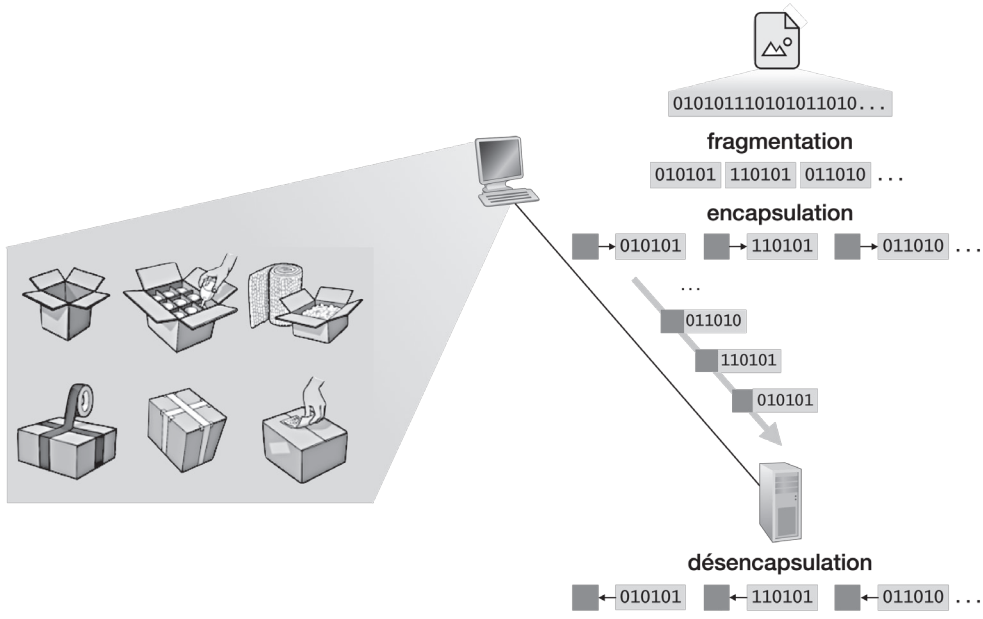

Figure 1.2. Fragmentation et encapsulation. Les bits de données appartenant à un même fichier sont fragmentés et répartis dans plusieurs messages. Un entête est ajouté à chaque fragment. Les entêtes contiennent des bits de contrôle nécessaires au transport des données. L'ajout et le retrait de l'entête sont appelés respectivement encapsulation et désencapsulation.

1.2 | Empaquetage des données

À la manière des produits ou denrées qu'un transporteur achemine sur le réseau routier ou par les airs, les données ont besoin d'être empaquetées. Comme le montre la Figure 1.2, les opérations de base qui permettent l'empaquetage des données sont la fragmentation et l'encapsulation.

1.2.1 Fragmentation

Dans un réseau de données, les données sont fragmentées : une longue série de bits de données appartenant à un même fichier est répartie dans plusieurs colis appelés messages. À la manière des dimensions ou du poids d'un colis, les messages sont soumis à des restrictions qui imposent une longueur minimale et maximale. La longueur d'un message est exprimée en nombre de bits ou d'octets.

1.2.2 Encapsulation et désencapsulation

L'équivalent des cartons d'emballage est une série de bits appelée entête, ajoutée en tête des bits de données. L'ajout d'un entête est appelé encapsulation et le retrait de l'entête, désencapsulation.

1.2.3 Entête et charge utile

Dans un réseau de données, un colis est donc un message constitué d'un entête et des bits de données. Les bits de données forment le corps du message que l'on appelle aussi charge utile (ou *payload*) du message. Les messages ayant une longueur maximale, plusieurs messages sont nécessaires pour transmettre un même objet tel qu'une image ou un fichier.

Dans l'entête d'un message, on retrouve des bits de contrôle tels que l'adresse du destinataire et celle de l'expéditeur. À la manière du papier-bulle utilisé pour éviter la casse des produits fragiles, l'entête peut également contenir des bits appelés somme de contrôle. La somme de contrôle permet de détecter les bits reçus en erreur. Une fois détectées, on peut réparer les erreurs en retransmettant les messages erronés.

Les bits correspondant aux adresses ou à la somme de contrôle sont des exemples de champ d'entête. Un entête est une succession de champs dont la longueur est variable ou fixe. La structure d'un entête aussi appelée format d'un entête, fait référence aux champs qui composent cet entête.

1.2.4 Format d'un entête

Un champ d'entête est défini par son nom, sa longueur exprimée en bits et sa signification selon les valeurs qu'il peut contenir. La valeur d'un champ peut être fournie sous la forme d'un code binaire : la valeur 0x06 du champ *Protocole* de l'entête ajouté par le protocole IP, fait référence au protocole TCP. D'autres champs peuvent contenir une valeur littérale représentée par son codage ASCII : la valeur 0x474554 du champ *Méthode* de l'entête HTTP indique la méthode de requête GET.

1.3 | Garanties de livraison

Comme nous le verrons dans la Section 1.4 de ce chapitre, plusieurs entêtes sont nécessaires afin d'assurer les nombreuses propriétés attendues concernant le transport des données, chaque entête étant ajouté dans un but précis.

De telles propriétés comprennent la fiabilité, l'efficacité ou la résistance au facteur d'échelle.

1.3.1 Fiabilité

La fiabilité (ou *reliability*) fait référence à la livraison des données sans perte et sans erreur. Des messages pouvant être perdus, reçus en erreur, en désordre ou dupliqués, des bits de contrôle sont inclus dans leur entête. Ces bits font référence à un numéro appelé numéro de séquence qui permet de détecter les messages manquants, de réordonner les messages reçus en désordre ou de supprimer les doublons.

On trouve également une somme de contrôle dont le rôle est de vérifier si le message contient des bits en erreur. Comme l'indique son nom, son calcul résulte de la somme des bits du message avant son envoi. À sa réception, le destinataire refait le même calcul et vérifie si le résultat coïncide avec la somme calculée par la source.

1.3.2 Efficacité

L'efficacité (ou *efficiency*) dans un réseau de données résulte de mécanismes similaires à ceux qu'un transporteur met en œuvre pour s'assurer que ses fourgons sont tous utilisés au maximum de leur capacité de transport. Les messages étant caractérisés par une longueur maximale, leur envoi est efficace si les bits de données qui occupent leur charge utile, sont en nombre suffisant pour que cette longueur soit atteinte.

L'efficacité peut également faire référence à la proportion du message occupée par son entête comparée à sa charge utile. Si envoyer un colis est facturé sur la base de son poids brut, l'expéditeur cherchera à minimiser le poids du carton d'emballage tout en maximisant celui de son contenu.

Il en va de même des autoroutes dont le nombre de voies est déterminé de façon à éviter les bouchons mais également leur sous-utilisation. Les coûts de réalisation et d'exploitation des autoroutes doivent être en adéquation avec leur utilisation, leur financement dépendant des recettes aux péages. On parle alors d'utilisation efficace des autoroutes.

De la même manière, les coûts liés au déploiement et à la maintenance d'un réseau de données ainsi que sa capacité, doivent être en adéquation avec le volume des données transportées, le nombre d'utilisateurs et le prix d'un abonnement Internet.

1.3.3 Passage à l'échelle

Une autre propriété des réseaux est le passage à l'échelle (ou *scalability*). Un système est capable de passer à l'échelle lorsque ses performances ne sont pas, ou sont peu sensibles à une grandeur dénombrable croissante, caractérisant ce système. Cette grandeur peut faire référence au nombre d'utilisateurs ou de nœuds dans un réseau. On parle aussi de résistance au facteur d'échelle.

Une autre grandeur peut aussi faire référence au nombre de messages en transit dans le réseau ou au nombre de requêtes que reçoit un serveur. Pour reprendre le parallèle avec le fret, un transporteur doit s'assurer du bon déroulement de ses opérations, y compris à l'occasion des fêtes de fin d'années.

Dans un réseau de données, pour assurer les nombreuses propriétés attendues concernant la livraison des données, un seul entête ne suffit pas.

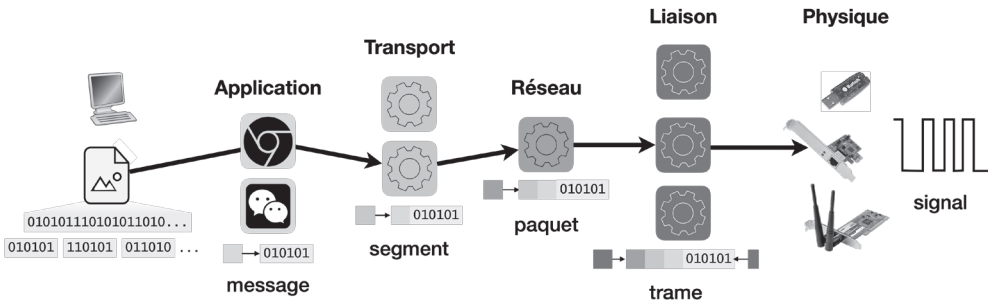

Figure 1.3. Encaspulsation et ajout d'entêtes. Préalablement à leur envoi, les données transitent le long d'une chaîne de programmes qui ajoutent chacun un entête. Le dernier programme à ajouter un entête ajoute également un enqueue.

1.4 | Architecture en couches

Un message comprend une série d'entêtes ajoutée aux bits de données. Un entête est ajouté dans un but spécifique qui détermine le type des informations contenues dans cet entête. L'ajout d'un entête se fait en sollicitant un programme qui prend en paramètre, les bits de données.

Les bits de données passent par une série de programmes qui se relaient pour ajouter un entête supplémentaire avant que le message final soit prêt à être transmis sur le support de transmission. Comme le montre la Figure 1.3, chaque programme est identifié par un nom qui dépend de sa position. Le premier programme est appelé Application. Il est suivi des programmes appelés Transport, Réseau, Liaison et Physique.

Le message résultant de l'ajout d'un entête est désigné par un nom particulier qui dépend du programme à avoir ajouté cet entête. Le message résultant de l'ajout d'entête au niveau Transport, est appelé segment.

Le message résultant de l'ajout de l'entête par le programme nommé Réseau est appelé paquet. Un paquet est un segment auquel le programme Transport ajoute un nouvel entête.

Le dernier programme à ajouter un entête, est appelé Liaison et a la particularité d'ajouter également des bits en fin du message appelés, de ce fait, enqueue. Le message résultant de l'ajout de l'entête et de l'enqueue par le programme Liaison, est appelé trame.

Le programme dénommé Physique est responsable de transformer la séquence des bits contenus dans une trame sous la forme d'un signal électromagnétique adapté au support de transmission. C'est le rôle de la carte réseau qui est le composant matériel responsable de transmettre (resp. de recevoir) les données sur un support de transmission auquel il est physiquement connecté.

À la réception du message final, les entêtes sont progressivement retirés en traversant une séquence de programmes similaire à celle traversée avant l'émission de ce message mais en sens inverse. Le dernier entête à avoir été ajouté, est le plus externe et est donc le plus accessible. Chaque programme accède uniquement à l'entête ajouté côté émetteur, par son programme homologue.

1.4.1 Couches et protocoles

Dans Internet, les programmes qui contribuent à la préparation des données en vue de leur transmission, sont au nombre de cinq (5). En première position, on retrouve les programmes applicatifs qui génèrent les données côté émetteur et les consomment côté récepteur. En dernière position, on retrouve la carte réseau (*Network interface card* ou NIC) dont les composants matériels transforment le message sous la forme d'un signal émis sur le support de transmission.

Couche. Par souci de représentation, ces programmes sont empilés les uns sur les autres et la traversée des données se fait de haut en bas côté émetteur et de bas en haut côté récepteur. On parle alors de couches en lieu et en place de ces programmes.

Une même couche peut être implémentée de plusieurs manières différentes, selon les propriétés attendues de cette couche. La Figure 1.4 représente l'empilement des cinq couches qui contribuent dans Internet, à la préparation des données côté émetteur, en vue de leur transmission. Les données traversent la séquence particulière des programmes dont la sélection au niveau de chaque couche, correspond au service attendu.

Protocole. Une couche collabore avec sa couche homologue distante selon des règles bien précises concernant la structure de l'entête qu'elle ajoute aux messages côté émetteur et retire côté récepteur. La traversée des couches est représentée dans la Figure 1.5.

Ces règles définissent également les actions à exécuter en vue de l'émission ou suite à la réception d'un message. De ces actions peuvent résulter l'installation, la mise à jour ou la suppression d'un état.

État. Un état fait référence à une valeur stockée en mémoire qui caractérise l'état de l'échange entre deux machines. On distingue les états durs (ou *hard state*) des états mous (ou *soft state*). Pour être supprimé, un état dur requiert une indication explicite telle que la réception d'un message.

Dans le cas d'un état mou, ce dernier est accompagné d'un temporisateur qui permet de limiter la durée de vie (*Time to live* ou TTL) de cet état. On parle aussi de l'âge (*Age*) d'un état. À moins que l'état soit rafraîchi, le temporisateur expirera, provoquant ainsi la suppression programmée de l'état associé.

L'ensemble de ces règles forme un protocole. Pour communiquer entre elles, deux machines exécutent simultanément plusieurs protocoles, un protocole par couche.

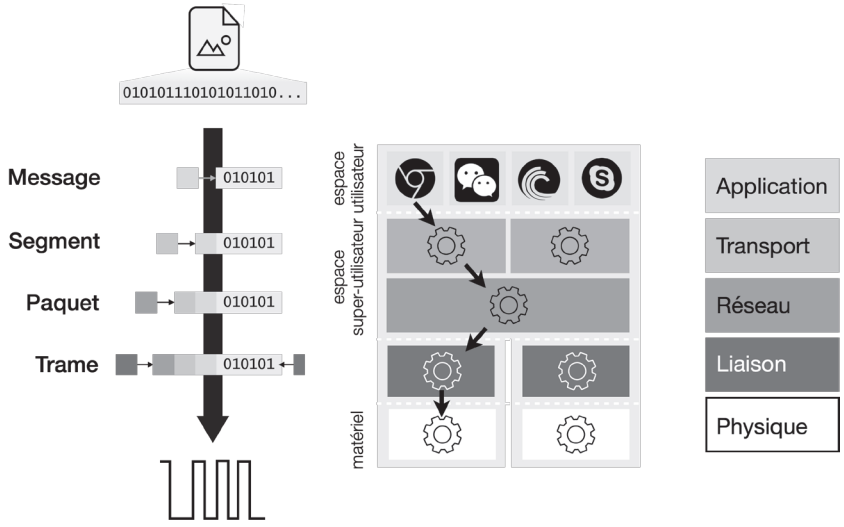

Figure 1.4. Pile protocolaire. Dans Internet, les couches qui contribuent à la préparation des données en vue de leur transmission, sont au nombre de cinq (5). Certaines couches pouvant offrir plusieurs types de service, les données traversent la séquence particulière de programmes correspondant au service attendu.

1.4.2 Pile protocolaire

L'empilement des couches et l'ensemble des protocoles qu'elles implémentent, forment une pile protocolaire. Dans le cas d'Internet, on parle de pile TCP/IP en référence aux deux protocoles TCP et IP implémentés respectivement par la couche Transport (4) et Réseau (3).

Standardisation. Certains protocoles ont été standardisés, rendant ainsi leurs spécifications accessibles à tous. C'est le cas par exemple de *Hypertext transfer protocol* (HTTP), le protocole qui régit les communications entre navigateurs et serveurs Web.

Les spécifications de HTTP étant publiques, plusieurs navigateurs sont proposés par différentes sociétés telles que Microsoft ou Google. Du fait d'avoir été développés conformément au standard, navigateurs et serveurs Web savent communiquer entre eux.

Entête de protocole. À la manière des étiquettes apposées sur un colis, un protocole ajoute un entête aux messages reçus de la couche immédiatement supérieure. Dans le cas des protocoles de la couche Application (7), l'entête est ajouté aux données passées par l'utilisateur. La position de cet entête est déterminée par la couche à laquelle le protocole appartient. L'entête est ajouté en vue d'être lu par sa couche homologue, une fois le message reçu.