Le but du protocole ARP

Le protocole **ARP** a un rôle phare parmi les protocoles de la couche Internet de la suite TCP/IP, car il permet de connaître l'adresse physique d'une carte réseau correspondant à une adresse IP, c'est pour cela qu'il s'appelle *Protocole de résolution d'adresse* (en anglais ARP signifie *Address Resolution Protocol*).

Chaque machine connectée au réseau possède un numéro d'identification de 48 bits. Ce numéro est un numéro unique qui est fixé dès la fabrication de la carte en usine. Toutefois la communication sur Internet ne se fait pas directement à partir de ce numéro (car il faudrait modifier l'adressage des ordinateurs à chaque fois que l'on change une carte réseau) mais à partir d'une adresse dite logique attribuée par un organisme: l'adresse IP.

Ainsi, pour faire correspondre les adresses physiques aux adresses logiques, le protocole ARP interroge les machines du réseau pour connaître leur adresse physique, puis crée une table de correspondance entre les adresses logiques et les adresses physiques dans une mémoire cache.

Lorsqu'une machine doit communiquer avec une autre, elle consulte la table de correspondance. Si jamais l'adresse demandée ne se trouve pas dans la table, le protocole ARP émet une requête sur le réseau. L'ensemble des machines du réseau vont comparer cette adresse logique à la leur. Si l'une d'entre-elles s'identifie à cette adresse, la machine va répondre à ARP qui va stocker le couple d'adresses dans la table de correspondance et la communication va alors pouvoir avoir lieu...

Le protocole RARP

Le protocole **RARP** (*Reverse Address Resolution Protocol*) est beaucoup moins utilisé, il signifie *Protocole ARP inversé*, il s'agit donc d'une sorte d'annuaire inversé des adresses logiques et physiques.
En réalité le protocole RARP est essentiellement utilisé pour les stations de travail n'ayant pas de disque dur et souhaitant connaître leur adresse physique...

Le protocole RARP permet à une station de connaître son adresse IP à partir d'une table de correspondance entre adresse MAC (adresse physique) et adresses IP hébergée par une passerelle (gateway) située sur le même réseau local (LAN).

Pour cela il faut que l'administrateur paramètre le gateway (routeur) avec la table de correspondance des adresses MAC/IP. En effet, à la différence de ARP ce protocole est statique. Il faut donc que la table de correspondance soit toujours à jour pour permettre la connexion de nouvelles cartes réseau.

RARP souffre de nombreuses limitations. Il nécessite beaucoup de temps d'administration pour maintenir des tables importantes dans les serveurs. Cela est d'autant plus vrai que le réseau est grand. Cela pose les problèmes de la ressource humaine, nécessaire au maintien des tables de correspondance et des capacités des matériels hébergeant la partie serveur du protocole RARP. En effet, RARP permet à plusieurs serveurs de répondre à des requêtes, bien qu'il ne prévoit pas de mécanismes garantissant que tous les serveurs soient capables de répondre, ni même qu'ils répondent de manière identique. Ainsi, dans ce type d'architecture on ne peut avoir confiance en un serveur RARP pour savoir si à une adresse MAC peut être

liée à une adresse IP parce que d'autres serveurs ARP peuvent avoir une réponse différente. Une autre limitation de RARP est qu'un serveur ne peut servir qu'un LAN.

Pour pallier les deux premiers problèmes d'administration, le protocole RARP peut être remplacé par le protocole DRARP, qui en est une version dynamique. Une autre approche, consiste à utiliser un serveur DHCP, qui lui, permet une résolution dynamique des adresses. De plus, DHCP est compatible avec le protocole BOOTP. Comme ce dernier il est routable ce qui permet de servir plusieurs LAN. Il ne marche qu'avec IP.

Plus d'informations

La principale documentation sur les protocole ARP et RARP est constituée par les RFCs :

- RFC 826 - ARP (An Ethernet Address Resolution Protocol)
- RFC 903 - RARP (Reverse Address Resolution Protocol)

La gestion des erreurs

Le protocole **ICMP** (*Internet Control Message Protocol*) est un protocole qui permet de gérer les informations relatives aux erreurs aux machines connectées. Etant donné le peu de contrôles que le protocole IP réalise, il permet non pas de corriger ces erreurs mais de faire part de ces erreurs aux protocoles des couches voisines. Ainsi, le protocole ICMP est utilisé par tous les routeurs, qui l'utilisent pour signaler une erreur (appelé *Delivery Problem*).

Les messages ICMP sont encapsulés

Les messages d'erreur ICMP sont transportés sur le réseau sous forme de datagramme, comme n'importe quelle donnée. Ainsi, les messages d'erreur peuvent eux-mêmes être sujet d'erreurs.

Toutefois en cas d'erreur sur un datagramme transportant un message ICMP, aucun message d'erreur n'est délivré pour éviter un effet "boule de neige" en cas d'incident sur le réseau.

Voici à quoi ressemble un message ICMP encapsulé dans un datagramme IP:

En-tête	Message ICMP			
	Type (8 bits)	Code (8 bits)	Checksum (16 bits)	Message (taille variable)

Signification des messages ICMP

Type	Code	Message	Signification du message
8	0	Demande d'ECHO	Ce message est utilisé lorsqu'on utilise la commande *PING*. Cette commande, permettant de tester le réseau, envoie un datagramme à un destinataire et lui demande de le restituer
3	0	Destinataire inaccessible	Le réseau n'est pas accessible
3	1	Destinataire inaccessible	La machine n'est pas accessible
3	2	Destinataire inaccessible	Le protocole n'est pas accessible
3	3	Destinataire inaccessible	Le port n'est pas accessible
3	4	Destinataire inaccessible	Fragmentation nécessaire mais impossible à cause du drapeau (flag) DF
3	5	Destinataire inaccessible	Le routage a échoué
3	6	Destinataire inaccessible	Réseau inconnu
3	7	Destinataire inaccessible	Machine inconnue
3	8	Destinataire inaccessible	Machine non connectée au réseau (inutilisé)
3	9	Destinataire inaccessible	Communication avec le réseau interdite
3	10	Destinataire inaccessible	Communication avec la machine interdite

3	11	Destinataire inaccessible	Réseau inaccessible pour ce service
3	12	Destinataire inaccessible	Machine inaccessible pour ce service
3	11	Destinataire inaccessible	Communication interdite (filtrage)
4	0	Source Quench	Le volume de données envoyé est trop important, le routeur envoie ce message pour prévenir qu'il sature afin de demander de réduire la vitesse de transmission
5	0	Redirection pour un hôte	Le routeur remarque que la route d'un ordinateur n'est pas optimale et envoie l'adresse du routeur à rajouter dans la table de routage de l'ordinateur
5	1	Redirection pour un hôte et un service donné	Le routeur remarque que la route d'un ordinateur n'est pas optimale pour un service donné et envoie l'adresse du routeur à rajouter dans la table de routage de l'ordinateur
5	2	Redirection pour un réseau	Le routeur remarque que la route d'un réseau entier n'est pas optimale et envoie l'adresse du routeur à rajouter dans la table de routage des ordinateurs du réseau
5	3	Redirection pour un réseau et un service donné	Le routeur remarque que la route d'un réseau entier n'est pas optimale pour un service donné et envoie l'adresse du routeur à rajouter dans la table de routage des ordinateurs du réseau
11	0	Temps dépassé	Ce message est envoyé lorsque le temps de vie d'un datagramme est dépassé. L'en-tête du datagramme est renvoyé pour que l'utilisateur sache quel datagramme a été détruit
11	1	Temps de ré-assemblage de fragment dépassé	Ce message est envoyé lorsque le temps de ré-assemblage des fragments d'un datagramme est dépassé.
12	0	En-tête erronné	Ce message est envoyé lorsqu'un champ d'un en-tête est erronné. La position de l'erreur est retournée
13	0	Timestamp request	Une machine demande à une autre son heure et sa date système (universelle)
14	0	Timestamp reply	La machine réceptrice donne son heure et sa date système afin que la machine émettrice puisse déterminer le temps de transfert des données
15	0	Demande d'adresse réseau	Ce message permet de demander au réseau une adresse IP
16	0	Réponse d'adresse réseau	Ce message répond au message précédent
17	0	Demande de masque de sous-réseau	Ce message permet de demander au réseau un masque de sous-réseau
18	0	Réponse de masque de sous-réseau	Ce message répond au message précédent
17	0	Timestamp reply	La machine réceptrice donne son heure et sa date système afin que la machine émettrice puisse déterminer le temps de transfert des données

Plus d'informations

Pour plus d'informations sur le protocole ICMP, le mieux est de se reporter à la RFC 792 expliquant de manière détaillée le protocole.

Les caractéristiques du protocole TCP

TCP (qui signifie *Transmission Control Protocol*, soit en français: *Protocole de Contrôle de Transmission*) est un des principaux protocoles de la couche transport du modèle TCP/IP. Il permet, au niveau des applications, de gérer les données en provenance (ou à destination) de la couche inférieure du modèle (c'est-à-dire le protocole IP). Lorsque les données sont fournies au protocole IP, celui-ci les encapsule dans des datagrammes IP, en fixant le champ protocole à 6 (Pour savoir que le protocole en amont est TCP...). TCP est un protocole orienté connexion, c'est-à-dire qu'il permet à deux machines qui communiquent de contrôler l'état de la transmission.
Les caractéristiques principales du protocole TCP sont les suivantes:

- TCP permet de remettre en ordre les datagrammes en provenance du protocole IP
- TCP permet de vérifier le flot de données afin d'éviter une saturation du réseau
- TCP permet de formater les données en segments de longueur variable afin de les "remettre" au protocole IP
- TCP permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne
- TCP permet enfin l'initialisation et la fin d'une communication de manière courtoise

Le but de TCP

Grâce au protocole TCP, les applications peuvent communiquer de façon sûre (grâce au système d'accusés de réception du protocole TCP), indépendamment des couches inférieures. Cela signifie que les routeurs (qui travaillent dans la couche Internet) ont pour seul rôle l'acheminement des données sous forme de datagrammes, sans se préoccuper du contrôle des données, car celui-ci est réalisé par la couche transport (plus particulièrement par le protocole TCP).

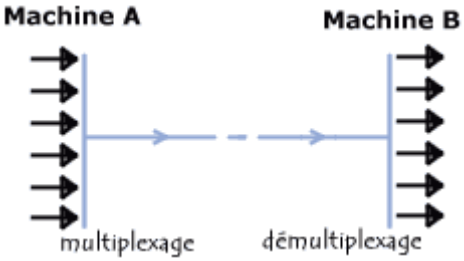
Lors d'une communication à travers le protocole TCP, les deux machines doivent établir une connexion. La machine émettrice (celle qui demande la connexion) est appelée client, tandis que la machine réceptrice est appelée serveur. On dit qu'on est alors dans un environnement Client-Serveur.
Les machines dans un tel environnement communiquent en mode connecté, c'est-à-dire que la communication se fait dans les deux sens.

Pour permettre le bon déroulement de la communication et de tous les contrôles qui l'accompagnent, les données sont encapsulées, c'est-à-dire qu'on ajoute aux paquets de données un en-tête qui va permettre de synchroniser les transmissions et d'assurer leur réception.

Une autre particularité de TCP est de pouvoir réguler le débit des données grâce à sa capacité à émettre des messages de taille variable, ces messages sont appelés *segments*.

La fonction de multiplexage

TCP permet d'effectuer une tâche importante: le multiplexage/démultiplexage, c'est-à-dire faire transiter sur une même ligne des données provenant d'applications diverses ou en d'autres mots mettre en série des informations arrivant en parallèle.



Ces opérations sont réalisées grâce au concept de ports (ou sockets), c'est-à-dire un numéro associé à un type d'application, qui, combiné à une adresse IP, permet de déterminer de façon unique une application qui tourne sur une machine donnée.

Le format des données sous TCP

Un segment TCP est constitué comme suit:

<table>
<tr><td colspan="16">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td><td colspan="16">16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td></tr>
<tr><td colspan="16">Port Source</td><td colspan="16">Port destination</td></tr>
<tr><td colspan="32">Numéro d'ordre</td></tr>
<tr><td colspan="32">Numéro d'accusé de réception</td></tr>
<tr><td colspan="4">Décalage données</td><td colspan="6">réservée</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Fenêtre</td></tr>
<tr><td colspan="16">Somme de contrôle</td><td colspan="16">Pointeur d'urgence</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Remplissage</td></tr>
<tr><td colspan="32">Données</td></tr>
</table>

Signification des différents champs:

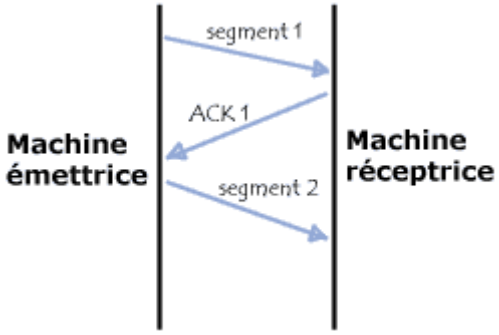
- **Port Source** (16 bits): Port relatif à l'application en cours sur la machine source
- **Port Destination** (16 bits): Port relatif à l'application en cours sur la machine de destination
- **Numéro d'ordre** (32 bits): Lorsque le drapeau SYN est à 0, le numéro d'ordre est celui du premier mot du segment en cours.
  Lorsque SYN est à 1, le numéro d'ordre est égal au numéro d'ordre initial utilisé pour synchroniser les numéros de séquence (ISN)
- **Numéro d'accusé de réception** (32 bits): Le numéro d'accusé de réception également appelé numéro d'acquittement correspond au numéro (d'ordre) du prochain segment attendu, et non le numéro du dernier segment reçu.
- **Décalage des données** (4 bits): il permet de repérer le début des données dans le paquet. Le décalage est ici essentiel car le champ d'options est de taille variable
- **Réservé** (6 bits): Champ inutilisé actuellement mais prévu pour l'avenir
- **Drapeaux (flags)** (6x1 bit): Les drapeaux représentent des informations supplémentaires:
  - **URG**: si ce drapeau est à 1 le paquet doit être traité de façon urgente.
  - **ACK**: si ce drapeau est à 1 le paquet est un accusé de réception.
  - **PSH** (PUSH): si ce drapeau est à 1, le paquet fonctionne suivant la méthode PUSH.
  - **RST**: si ce drapeau est à 1, la connexion est réinitialisée.
  - **SYN**: Le Flag TCP SYN indique une demande d'établissement de connexion.
  - **FIN**: si ce drapeau est à 1 la connexion s'interrompt.

- **Fenêtre** (16 bits): Champ permettant de connaître le nombre d'octets que le récepteur souhaite recevoir sans accusé de réception
- **Somme de contrôle** (Checksum ou CRC): La somme de contrôle est réalisée en faisant la somme des champs de données de l'en-tête, afin de pouvoir vérifier l'intégrité de l'en-tête
- **Pointeur d'urgence** (16 bits): Indique le numéro d'ordre à partir duquel l'information devient urgente
- **Options** (Taille variable): Des options diverses
- **Remplissage**: On remplit l'espace restant après les options avec des zéros pour avoir une longueur multiple de 32 bits

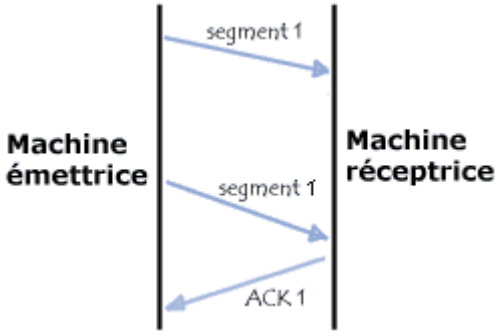
Fiabilité des transferts

Le protocole TCP permet d'assurer le transfert des données de façon fiable, bien qu'il utilise le protocole IP, qui n'intègre aucun contrôle de livraison de datagramme.

En réalité, le protocole TCP possède un système d'accusé de réception permettant au client et au serveur de s'assurer de la bonne réception mutuelle des données.
Lors de l'émission d'un segment, un **numéro d'ordre** (appelé aussi *numéro de séquence*) est associé. A réception d'un segment de donnée, la machine réceptrice va retourner un segment de donnée dont le drapeau ACK est à 1 (afin de signaler qu'il s'agit d'un accusé de réception) accompagné d'un numéro d'accusé de réception égal au numéro d'ordre précédent.



De plus, grâce à une minuterie déclenchée dès réception d'un segment au niveau de la machine émettrice, le segment est réexpédié dès que le temps imparti est écoulé, car dans ce cas la machine émettrice considère que le segment est perdu...



Toutefois, si le segment n'est pas perdu et qu'il arrive tout de même à destination, la machine réceptrice saura grâce au numéro d'ordre qu'il s'agit d'un doublon et ne conservera que le dernier segment arrivé à destination...

Etablissement d'une connexion

Etant donné que ce processus de communication, qui se fait grâce à une émission de données et d'un accusé de réception, est basé sur un numéro d'ordre (appelé généralement *numéro de séquence*), il faut que les machines émettrices et réceptrices (client et serveur) connaissent le numéro d'ordre initial de l'autre machine.

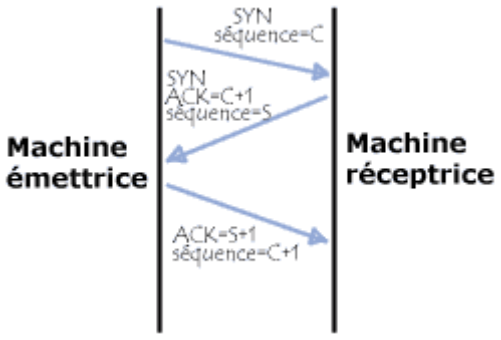
L'établissement de la connexion entre deux applications se fait souvent selon le schéma suivant:

- Les ports TCP doivent être ouverts
- L'application sur le serveur est passive, c'est-à-dire que l'application est à l'écoute, en attente d'une connexion
- L'application sur le client fait une requête de connexion sur le serveur dont l'application est en ouverture passive. L'application du client est dite "en ouverture active"

Les deux machines doivent donc synchroniser leurs séquences grâce à un mécanisme communément appelé *three ways handshake* (*poignée de main en trois temps*), que l'on retrouve aussi lors de la clôture de session.

Ce dialogue permet d'initier la communication, il se déroule en trois temps, comme sa dénomination l'indique:

- Dans un premier temps la machine émettrice (le client) transmet un segment dont le drapeau SYN est à 1 (pour signaler qu'il s'agit d'un segment de synchronisation), avec un numéro d'ordre N, que l'on appelle numéro d'ordre initial du client
- Dans un second temps la machine réceptrice (le serveur) reçoit le segment initial provenant du client, puis lui envoie un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1 et le drapeau SYN est à 1 (car il s'agit là encore d'une synchronisation). Ce segment contient le numéro d'ordre de cette machine (du serveur) qui est le numéro d'ordre initial du client. Le champ le plus important de ce segment est le champ accusé de réception qui contient le numéro d'ordre initial du client, incrémenté de 1
- Enfin, le client transmet au serveur un accusé de réception, c'est-à-dire un segment dont le drapeau ACK est à 1, dont le drapeau SYN est à zéro (il ne s'agit plus d'un segment de synchronisation). Son numéro d'ordre est incrémenté et le numéro d'accusé de réception représente le numéro d'ordre initial du serveur incrémenté de 1



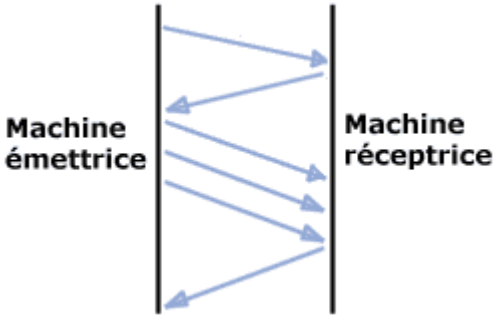
Suite à cette séquence comportant trois échanges les deux machines sont synchronisées et la communication peut commencer!

Il existe une technique de piratage, appelée spoofing IP, permettant de corrompre cette relation d'approbation à des fins malicieuses !

Méthode de la fenêtre glissante

Dans de nombreux cas, il est possible de limiter le nombre d'accusés de réception, afin de désengorger le réseau, en fixant un nombre de séquence au bout duquel un accusé de réception est nécessaire. Ce nombre est en fait stocké dans le champ *fenêtre* de l'en-tête TCP/IP.

On appelle effectivement cette méthode "*méthode de la fenêtre glissante*" car on définit en quelque sorte une fourchette de séquences n'ayant pas besoin d'accusé de réception, et celle-ci se déplace au fur et à mesure que les accusés de réception sont reçus.



De plus, la taille de cette fenêtre n'est pas fixe. En effet, le serveur peut inclure dans ses accusés de réception en stockant dans le champ fenêtre la taille de la fenêtre qui lui semble la plus adaptée. Ainsi, lorsque l'accusé de réception indique une demande d'augmentation de la fenêtre, le client va déplacer le bord droit de la fenêtre.

Par contre, dans le cas d'une diminution, le client ne va pas déplacer le bord droit de la fenêtre vers la gauche mais attendre que le bord gauche avance (avec l'arrivée des accusés de réception).

Fin d'une connexion

Le client peut demander à mettre fin à une connexion au même titre que le serveur.
La fin de la connexion se fait de la manière suivante:

- Une des machines envoie un segment avec le drapeau *FIN* à 1, et l'application se met en état d'attente de fin, c'est-à-dire qu'elle finit de recevoir le segment en cours et ignore les suivants
- Après réception de ce segment, l'autre machine envoie un accusé de réception avec le drapeau *FIN* à 1 et continue d'expédier les segments en cours. Suite à cela la machine informe l'application qu'un segment *FIN* a été reçu, puis envoie un segment *FIN* à l'autre machine, ce qui clôture la connexion...

Plus d'informations

Pour plus d'informations sur le protocole TCP, le mieux est de se reporter à la RFC 793 expliquant de manière détaillée le protocole.

Les caractéristiques du protocole UDP

Le protocole UDP (*User Datagram Protocol*) est un protocole non orienté connexion de la couche transport du modèle TCP/IP. Ce protocole est très simple étant donné qu'il ne fournit pas de contrôle d'erreurs (il n'est pas orienté connexion...).

L'en-tête du segment UDP est donc très simple:

Port Source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)
Données (longueur variable)	

Signification des différents champs

- **Port Source**: il s'agit du numéro de port correspondant à l'application émettrice du segment UDP. Ce champ représente une adresse de réponse pour le destinataire. Ainsi, ce champ est optionnel, cela signifie que si l'on ne précise pas le port source, les 16 bits de ce champ seront mis à zéro, auquel cas le destinataire ne pourra pas répondre (cela n'est pas forcément nécessaire, notamment pour des messages unidirectionnels.
- **Port Destination**: Ce champ contient le port correspondant à l'application de la machine destinataire à laquelle on s'adresse.
- **Longueur**: Ce champ précise la longueur totale du segment, en-tête comprise, or l'en-tête a une longueur de 4 x 16 bits (soient 8 x 8 bits) donc le champ longueur est nécessairement supérieur ou égal à 8 octets.
- **Somme de contrôle**: Il s'agit d'une somme de contrôle réalisée de telle façon à pouvoir contrôler l'intégrité du segment.

Plus d'informations

Pour plus d'informations sur le protocole UDP, le mieux est de se reporter à la RFC 768 expliquant de manière détaillée le protocole.