

TP N° 2 de Réseaux

Etude des protocoles ARP et ICMP

Pascal Sicard

1 INTRODUCTION

L'objectif de ce TP est d'observer et comprendre le protocole de résolution d'adresse ARP, et un protocole annexe : ICMP. Nous étudierons aussi en fin de TP le mécanisme de fragmentation du protocole IP.

Rappel : Les protocoles **ARP** (Address Resolution), **RARP** (Protocol/Reverse Address Resolution Protocol), **IP** (Internet Protocol) et **ICMP** (Internet Control Message Protocol) sont associés à la couche 3.

Les protocoles **TCP** (Transmission Control Protocol) et **UDP** (User Datagram Protocol) sont associés à la couche 4. La figure 1 montre les diverses interactions qui existent entre les protocoles d'Internet. Les traits représentent les encapsulations successives entre les différents PDU (Protocol Data Unit).

1.0.1 Le protocole de la couche 2

C'est le protocole CSMA/CD (norme ISO 802.3) qui est utilisé dans les cartes Ethernets qui sont sur les ordinateurs (voir séance précédente).

1.0.2 Protocoles de la couche 3

Le protocole ARP Le protocole ARP (Address Resolution Protocol) permet à une machine A de trouver, si elle existe, l'adresse Ethernet d'une autre machine B connectée sur le même réseau **local**, en donnant uniquement l'adresse Internet de celle-ci. Le but de ce protocole est de cacher aux applications opérant au niveau supérieur l'adresse physique des machines et de ne leur permettre de manipuler que les adresses Internet. L'utilisateur n'a pas directement accès à ARP : il est utilisé par le protocole IP, quand cela est nécessaire.

Remarque : Le protocole RARP (Reverse-ARP) permet de trouver l'adresse INTERNET d'une machine du réseau à partir de son adresse Ethernet (intéressant pour « booter » une machine via le réseau).

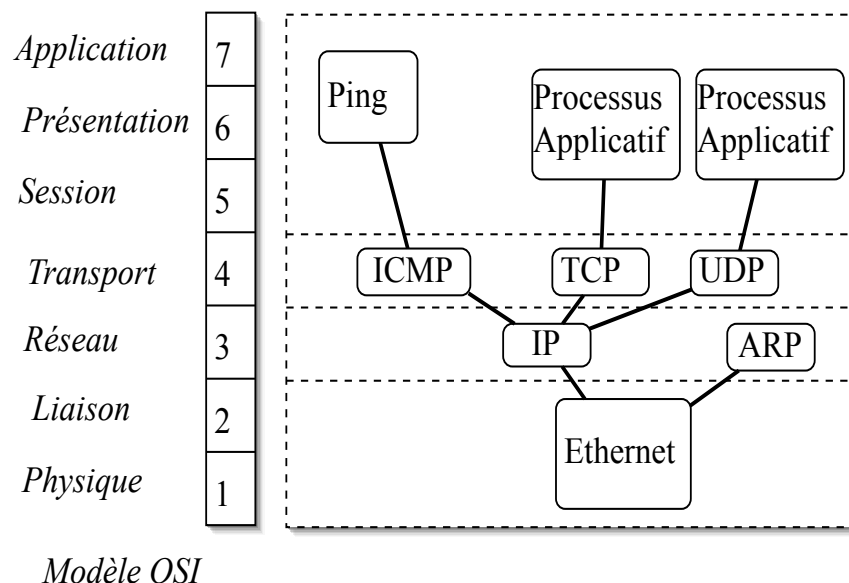


FIGURE 1 – Principaux Protocoles

Le Protocole IP Pour bien comprendre la problématique résolue par les protocoles de la couche 3, il faut bien avoir à l'esprit la nuance qu'il existe entre un réseau local (LAN) et un réseau étendu (WAN), les deux étant souvent appelé "réseau". Un réseau local est un réseau regroupant un ensemble de machines peu éloignées géographiquement (par exemple un réseau Ethernet). Un réseau étendu permet de relier des réseaux locaux à l'aide de liens « longues distances » (voir Fig 2).

Ainsi, le protocole Internet permet aux couches supérieures de faire abstraction de l'ensemble des réseaux (locaux et étendus) qu'il faut parfois traverser pour acheminer un paquet dans le réseau. Ainsi les couches supérieures n'ont pas à se soucier de la route parfois compliquée que les paquets doivent suivre, elles voient la liaison comme une liaison directe entre la machine émettrice et la machine réceptrice. Ce problème de gestion des routes à prendre dans un "réseau étendu" est appelé **routing**. D'autre part, ces différents réseaux composant le "réseau étendu" peuvent être hétérogènes, c'est-à-dire utiliser des protocoles différents (au niveau 2) et des trames de différentes longueurs. Le découpage et le réassemblage des paquets (appelé mécanisme de **fragmentation**) est aussi résolu par ce protocole de la couche 3.

Le service de base offert par IP est l'émission et la réception de paquets de données appelés datagrammes. Ce service est dit "non fiable" dans la mesure où la perte (ou l'altération) d'un paquet pendant son transport n'est pas résolue. Aucun mécanisme permettant de récupérer ces erreurs n'existe dans la couche IP.

Le Protocole ICMP Nous avons vu que IP offre un service non fiable, c'est-à-dire que si un paquet est perdu ou qu'une anomalie quelconque se produit au niveau des fonctionnalités de IP, celui ci ne rapporte aucune information quant à l'occurrence, la nature ou l'origine d'une telle erreur.

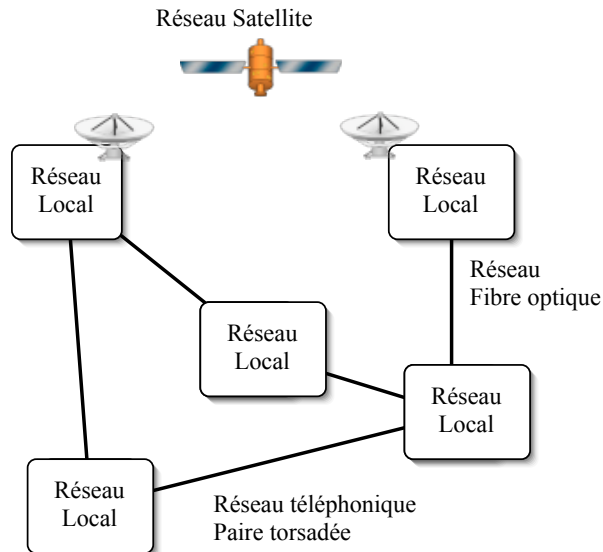


FIGURE 2 – Réseaux locaux interconnectés par des réseaux longue distance

Pour remédier à cette faiblesse du protocole, les concepteurs ont introduit dans la famille de protocoles TCP/IP un protocole appelé ICMP. La fonctionnalité principale de ICMP est de rapporter, à la station émettrice du paquet, les erreurs qui peuvent se produire au niveau IP.

Ainsi, quand le protocole IP n'arrive pas, dans certains cas, à remplir correctement la tâche qui lui est assignée, il l'indique au protocole ICMP qui émet un paquet à destination de la station source notifiant la nature et l'origine de l'erreur. Ce paquet est récupéré par le protocole ICMP de la station source qui informe le protocole IP de l'occurrence de cette erreur. Le protocole IP ainsi avisé peut agir en conséquence.

Remarque : ce protocole est associé à la couche 3 mais il est lui-même encapsulé par IP comme s'il se situait dans la couche 4 (voir figure 1).

2 DEROULEMENT DU TP :



: Cette icône indique par la suite les expérimentations à effectuer et à résumer.



: Celle ci indique les questions auxquelles il faut donner une réponse précise et détaillée dans votre compte rendu.

Dans ce TP, au moins 3 ordinateurs doivent être connectés sur un même réseau.

IMPORTANT : Remplissez les fichiers `/etc/hosts` sur les machines.

Les captures et les analyses des paquets générés par les manipulations se font à l'aide de l'utilitaire **Wireshark** (cf. doc). Ces captures pourront se faire sur l'une quelconques des 3 machines.

2.1 Etude des protocoles de niveau 3

2.1.1 Le protocole ARP

Syntaxe des paquets ARP Le but de la manipulation suivante est de générer et d'analyser des paquets de type ARP. Il arrive parfois, comme vous l'avez constaté dans le premier TP, que les paquets ARP ne soient pas systématiquement générés parce que les correspondances entre adresse Internet et Ethernet sont conservées dans une table que le protocole ARP met à jour périodiquement.

Dans certaines étapes des manipulations suivantes, on utilisera la commande **arp -a -d** pour vider le contenu de cette table. Si la commande **arp** a un temps de réponse long, remplissez le fichier `/etc/hosts` cela devrait résoudre le problème



Opérations :

- Sur un premier ordinateur, exécutez **arp -a -d**, puis, à l'aide de **ping**, vérifiez la présence d'un second ordinateur du réseau.
- Lorsque deux paquets ARP ont été capturés, arrêtez **Wireshark**.
- Retrouvez dans le format hexadécimal, les valeurs des différents champs (entête ETHER et données ARP) des paquets.





– 1 Qu'est ce qui permet d'identifier les paquets comme étant de type ARP ?

Format de la trame Ethernet : (entre parenthèses est indiqué le nombre d'octets des champs)

| | | | | | |
|---------------------------------------|----------------------|-----------------|-------------|---------------------------------|------------|
| Préambule (7) + marqueur début (1) | Destination @ (6) | Source @ (6) | Type (2) | Data (46-1500) | CRC (4) |
|---------------------------------------|----------------------|-----------------|-------------|---------------------------------|------------|


 — **2** Où se situe le paquet ARP dans la trame Ethernet ?

 — **3** De la même manière, faites le schéma correspondant au paquet ARP. Indiquez le rôle de chacun des champs du paquet ARP.


 — **4** Comment le niveau Ethernet connaît-il la taille des paquets qu'il reçoit ? Comment détermine-t-il la fin du paquet ?

Retrouvez les octets de bourrage d'Ethernet dans le paquet.

Attention les octets de bourrage d'Ethernet n'apparaissent pas si le paquet est capturé sur la station émettrice.

 — **5** Qu'est ce qui est transmis à la couche ARP par la couche Ethernet (à la réception d'un paquet) ? Ethernet connaît-il l'existence d'un bourrage à la réception d'un paquet.

Algorithme du protocole ARP Le but de cette section est de découvrir l'algorithme général du protocole ARP. Au cours des opérations suivantes, vous serez amenés à manipuler manuellement le contenu des tables du protocole ARP. La commande à utiliser est **arp** (cf doc).

 — **6** Votre tâche consiste à partir des manipulations et des observations détaillées plus loin, à compléter de façon informelle le corps de l'algorithme suivant :
tantque vrai **faire**

attendre(événement)

- **si** événement est "Question sur adresse internet" (requête interne de IP vers ARP)
...
fin**si**
- **si** événement est expiration du timer d'effacement associé à une entrée
...
fin**si**
- **si** événement est réception requête (ARP Request)

...

finsi

- ... (attention il y a d'autres événements)

fin tantque

Vous allez réaliser des manipulations sur les trois ordinateurs libres de la plate-forme (un ordinateur est réservé à la capture des paquets). Dans la suite, nous identifierons ces ordinateurs par A, B et C.

1.



Exécutez sur A un ping vers B, et notez les paquets engendrés.



– **7** Observez les tables ARP de A et B. Pourquoi B apparaît-il dans la table de A ? Comment B a-t-il pu apprendre l'adresse de A alors qu'il n'a pas envoyé de *ARP request* à A ?



Dans la table de B, supprimez l'entrée pour la station A.



– **8** Recommencez le ping de A vers B. Notez les paquets engendrés. Expliquez.



Un timer d'effacement est associé à chaque ligne de la table ARP, il est plutôt long. Vérifiez sa durée en consultant la table ARP juste après l'apprentissage d'une adresse.

N'attendez pas pour voir son effet !


2.



Dans la table de A, supprimez l'entrée pour la station B. Lancez l'outil de capture réseau. Faites un ping de A vers B que vous aurez au préalable débranché du réseau.



– **9** Est-ce que c'est le timer de ARP ou de Ping qui déclenche les re-émissions des paquets *ARP Request* ? Quelle est la durée de ce timer de re-émission ?

 – **10** Observez le réseau au moment où vous configurez une interface (*ifconfig*). A quoi peut servir le *ARP gratuit* ?


3.


 Videz les tables de A et B. Dans la table de A, ajoutez une entrée pour l'adresse IP de B, mais en précisant l'option **pub** (**pub** signifie "**published**") à la fin de la commande : **arp -s AdresseIP AdresseEthernet pub**.


Vérifiez que cette entrée est bien *Publish* quand vous demandez le contenu de la table par **arp -a**.

Observez l'activité sur le réseau au moment où vous rajoutez cette entrée dans la table ARP avec l'option PUB.

Videz la table ARP de C. Exécutez sur C un ping vers B. Notez à nouveau, les paquets engendrés, et le contenu de la table de A, B et C.


 – **11** Pourquoi deux paquets de type **ARP reply** sont-ils observés ? Expliquez ce qu'il s'est passé (demandez une analyse détaillée de ces paquets ; étudiez en particulier les adresses Ethernet et Internet).

 Videz la table ARP de A puis ajoutez dans A en PUB l'adresse IP de B mais avec l'adresse Ethernet de A. Que se passe-t-il alors lors d'un ping vers B depuis C ?

 – **12** Dans Dans quels cas la déclaration d'une adresse en *Publish* d'une autre machine peut-elle être intéressante ? (avec une autre adresse Ethernet par exemple)

4. **Question intéressante** : Que se passe-t-il si deux machines possèdent la même adresse Internet sur un réseau local ?

 Essayez.

 – **13** Au moment où vous configurez la deuxième machine, vous devriez avoir un message du système à l'écran (si vous sortez de l'environnement "*xinit*") sur la première. A quoi sert-il ? Qu'est-ce qui l'a déclenché ?



– 14 Expliquez ce qu’il se passe quand on fait un *ping* vers l’adresse IP commune ? Conclusions ?

Remarque : vous pouvez réaliser d’autres manipulations, afin de vérifier la validité de vos conclusions et compléter l’algorithme du protocole ARP.

2.1.2 Le protocole ICMP

Il existe une douzaine de types de messages ICMP. Voici les types de message les plus couramment utilisés.

- **ECHO REQUEST & ECHO REPLY** sont utilisés pour voir si une destination donnée (une station) est accessible et fonctionne. A la réception d’un message ECHO REQUEST, le destinataire doit répondre par un message ECHO REPLY.
- **DESTINATION UNREACHABLE** est généré quand le protocole IP ne sait pas comment joindre la station à qui est destiné un datagramme. Par exemple quand il n’a pas de réponse du protocole ARP.
- **SOURCE QUENCH** est utilisé pour ”brider” les stations qui envoient un trop grand nombre de datagrammes. A la réception de ce message, la station devrait modérer sa cadence d’émission des datagrammes.
- **TIME EXCEEDED** est envoyé lorsqu’un datagramme à son compteur (Time To Live) à zéro et doit donc être détruit.
- **PROBLEM PARAMETER** indique qu’une valeur illégale a été détectée dans un champ de l’entête d’un datagramme.

Le format du header ICMP varie suivant le type de message qu’il véhicule. Nous étudions dans cette section un exemple de messages ICMP qui sont utilisés par la commande ping (ECHO REQUEST et ECHO REPLY).

Bien qu’ils soient différents, les paquets ICMP possèdent tous un premier champ de 2 octets qui identifie le type et le code du message contenu dans le paquet. Le type 8 identifie par exemple un ECHO REQUEST et le type 0 un ECHO REPLY. Dans les deux cas, le code est à 0.

Le format du **paquet ICMP** est donné dans la figure 3.

Les champs **IDENTIFIER** et **SEQUENCE NUMBER** permettent uniquement de faire correspondre la réponse à une requête donnée.


Pour envoyer un paquet ICMP, vous aurez à remplir un fichier avec les valeurs correspondantes aux différents champs du paquet ICMP. Ensuite vous utiliserez le logiciel `send_icmp` pour envoyer le paquet :

```
# send icmp <host> <nom de fichier>
```


| | | |
|-----------------|----------|---------------------|
| Type (1) | Code (1) | Checksum (2) |
| Identifiant (2) | | Séquence Number (2) |
| Data optionnel | | |

FIGURE 3 – Format du paquet ICMP


où <nom de fichier> est le nom du fichier que vous avez créé et <host> est la station destination.


 Dans un fichier, écrivez en décimal ou hexadécimal (dans ce cas ajoutez x avant la valeur) les valeurs des différents octets correspondants au paquet ICMP. Les valeurs de chaque octet doivent être séparées par des blancs ou des passages à la ligne.

Remarque : Le calcul du checksum nécessite que la valeur initiale du champ checksum soit nulle.

Envoyez le paquet à l'aide de **send_icmp** et observez ce qu'il se passe à l'aide de Wireshark. Si vous n'avez pas reçu de réponse, vérifiez le format de votre paquet et refaites la manipulation.

Demandez une analyse hexadécimale des paquets **request** et **reply**.

 – **15** Que remarquez vous pour le champ **Identifiant** et le champ **Séquence Number** ? Observez ces deux champs dans les paquets générés par un ping ? A quoi servent ils ?

 – **16** Donnez l'algorithme de calcul des deux octets de Checksum. Vérifiez sur un de vos paquets ICMP capturés ce calcul. Quels types d'erreurs ce type de vérification peut-elle détecter et ne pas détecter ?

 Pour vérifier vos conclusions, refaites la manipulation en construisant des paquets différents.

2.2 Etude du protocole DHCP



Opérations

- Connectez un PC dans une prise murale via l'interface **em0**.
- Lancez une capture de paquet sur cette interface.
- Lancez la commande **dhclient em0** sur le PC.



– **17** Observez et commentez les paquets capturés alors. Expliquez en particulier les informations contenues dans le paquet DHCP ACK.

Aidez vous du *man* de *dhclient* et du cours pour expliquer le contenu de ces paquets.

Expliquez la nouvelle configuration de l'interface *em0*.

2.3 Etude de la fragmentation des paquets IP

Dans cette dernière manipulation, nous vous proposons d'aborder un aspect particulier du protocole IP (version 4) : la **fragmentation**.

La fragmentation est un mécanisme géré par le protocole IP lorsque le paquet traité est trop "gros" pour être directement envoyé sur le réseau (par exemple un réseau Ethernet ne peut pas traiter des paquets de taille supérieure à 1518 octets (entête Ethernet comprise). Dans ce cas, le paquet doit être fragmenté en plusieurs paquets de tailles plus petites. Quand un paquet a été fragmenté lors de son émission, il doit évidemment être réassemblé lors de sa réception (toujours au niveau IP).

Les champs **identification**, **flags** et **fragment offset** permettent à IP de gérer la fragmentation et le réassemblage.

Remarque : Le flag **DF** (Don't Fragment) permet d'interdire la fragmentation dans les routeurs intermédiaires. Quand il est positionné à 1, la fragmentation est interdite pour les routeurs suivants. Il se peut alors qu'il soit impossible d'acheminer un paquet. Le paquet est alors détruit et un paquet ICMP est envoyé à la source.

Le format de l'entête IP est donnée dans la figure 4. La longueur en bit des différents champs est donnée entre parenthèse.

| | | | | |
|-----------------------------|---------|------------------|---------------------|----------------------|
| Version(4) | Hlen(4) | Service Type (8) | Total length (16) | |
| Identification (16) | | | Flags(3) | Fragment Offset (13) |
| TimeToLive (8) | | Protocol (8) | HeaderChecksum (16) | |
| Source IP Address (32) | | | | |
| Destination IP Address (32) | | | | |
| IP options | | | Padding | |

FIGURE 4 – Format de l'entête IP

Sémantique des champs :

- **VERSION** représente le numéro de version du protocole IP utilisé. Ce champ est nécessaire pour vérifier que l'émetteur et le récepteur et toute machine intermédiaire sont en phase, c'est à dire utilisent le même format de datagramme.
- **HLEN** donne la longueur du header IP en mots de 32 bits. Cette valeur dépend de la longueur des champs IP_OPTION et PADDING qui est variable. SERVICE TYPE est un champ qui apporte des informations sur la façon dont le datagramme doit être traité pour répondre à des qualités de services particulières.
- **TOTAL LENGTH** donne la longueur totale du datagramme (en octets), entête IP et données réunis.
- **IDENTIFICATION, FLAGS et FRAGMENT OFFSET** : Ces trois champs interviennent dans le **mécanisme de fragmentation**.
- **TIME TO LIVE** représente le temps maximum que doit passer un paquet sur le réseau. Il est décrémenté régulièrement. Une fois le compteur à zéro le paquet est détruit.
- **PROTOCOL** spécifie le type des paquets (TCP, ICMP, UDP) encapsulé dans la trame IP. Il permet de délivrer le paquet au protocole de la couche supérieure spécifié.
- **HEADER CHECKSUM** est un code utilisé pour détecter les erreurs dans le datagramme (c'est-à-dire pour vérifier que ce datagramme n'a subi aucune altération).
- **SOURCE IP ADDRESS et DESTINATION IP ADDRESS** contiennent les adresses Internet des stations émettrice et réceptrice du datagramme. **IP OPTIONS** est un champ de taille variable (vide par défaut) utilisé généralement pour tester ou déboguer un réseau.
- **PADDING** est utilisé pour compléter le champ IP OPTIONS par des bits de bourrage afin que la taille du header soit un multiple de 32 bits.

Expérimentations :



Sur la station d'observation, lancez **Wireshark**.

Sur une autre station, réalisez l'émission d'un paquet UDP de 4600 octets à destination d'un port que vous aurez précédemment créé (Outils *udpmnt -s taille-paquet -n nombre-paquet* et *udptarget*).



— **18** Analysez les paquets engendrés sur le réseau. En particulier, étudiez les entêtes IP et expliquez les rôles des champs IDENTIFICATION, TOTAL LENGTH, FRAGMENT OFFSET, et du flag MF ("More Fragment"). Répéter si nécessaire la manipulation avec d'autres tailles de paquets.

Pourquoi l'entête UDP n'apparaît que dans un seul paquet ?

Dans quel ordre sont envoyés les fragments ?

Retrouvez dans l'hexadécimal des paquets l'entête UDP. L'entête UDP est elle dans le premier ou le dernier fragment ?

Retrouvez précisément dans ces paquets les 4600 octets de données de l'application. Faites une figure pour expliquer le découpage effectué par IP.