Cheick CISSOKO M2 Pro CCI

Gaëtan LAGIER

Eric THIERRY

**CR réseau TP2**

**LE PROTOCOLE ARP**

1) Ce qui permet d’identifier les paquets comme étant de type ARP est le champ “type” de la trame Ethernet. En effet lors de la capture de paquet avec l’application Wireshark, il est indiqué Type : ARP.

2) Dans la trame Ethernet, le paquet ARP se situe dans le champ Data.

3) Format de la trame ARP :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hardware  Type (2bytes) | Protocol  Type  (2bytes) | Hardware  Size (1byte) | Protocol  Size  (1byte) | Opcode  (2bytes) | @ Mac Source  (6bytes) | @ IP  Source  (4bytes) | @ Mac Destination  (6bytes) | @ IP  Destination  (4bytes) |

* Les champs Hardware (type t1 et size s1) et Protocole (type t2 et size s2) représentent respectivement le type t1 de taille s1 du protocole dont on veut la résolution en un autre type t2 de taille s2. Ces champs ne sont plus réellement utilisés aujourd’hui. En effet, le protocole ARP est exclusivement utilisé pour résoudre des adresses Ethernet en adresse IP, mais pour être universel et être utilisable pour d’autre type de taille différente, il a été crée ainsi.
* Le champ Opcode permet de préciser si le paquet est une requête ou une réponse : Request (x01), Reply (x02).
* Les champs restants sont respectivement l’adresse Ethernet et l’adresse IP de celui qui envoi le paquet, puis l’adresse Ethernet et l’adresse IP de celui qui est le destinataire de ce paquet.

4) Le niveau Ethernet ne connaît pas la taille des paquets. Il ne peut donc pas déterminer la fin du paquet.

RMQ : Les octets de bourrage correspondent à la parie padding située après le protocole ARP.

5) A la réception du paquet, toute la partie data est transmise à la couche ARP par la couche Ethernet. Comme le protocole Ethernet ne connaît pas la taille du paquet, il transmet la partie data + le padding qu’il considère comme faisant partie des données. Il n’a donc pas connaissance de ce bourrage.

6) **Algorithme du protocole ARP**:

Tant que (vrai) faire :

Attendre (évènement) :

* **Si** évènement est « Question sur adresse Internet » (requête interne de IP ver ARP), consulter si la table ARP contient l’adresse Ethernet de l’adresse IP ciblée. Si la table contient l’adresse, celle ci est transmise au protocole IP, sinon, un paquet ARP request est envoyé pour trouver l’adresse demandé par IP. **Finsi**
* **Si** évènement est expiration du timer d’effacement associé à une entrée, un paquet ARP request est envoyé pour trouver l’association de IP et Ethernet. **Finsi**
* **Si** événement est réception requête, envoi d’un paquet ARP Reply avec l’adresse Ethernet correspondant à l’adresse IP de la machine. **Finsi**
* **Si** événement est réception requête qui demande l’adresse Ethernet d’une adresse IP présente dans la table ARP en « Published », envois d’un paquet ARP avec l’association demandée. **Finsi**
* **Si** événement est réception d’un paquet ARP gratuit request : Soit l’adresse IP source de ce paquet est inconnu, dans ce cas aucune action n’est entreprise. Soit l’adresse IP source de ce paquet correspond à l’adresse IP de la machine courante, dans ce cas un message est affiché à l’utilisateur pour prévenir de la tentative de s’approprier l’adresse IP et un ARP gratuit reply est envoyé à destination de la machine émettrice de ce paquet pour prévenir que l’adresse IP qu’elle utilise est déjà utilisée par la machine courante. Soit l’adresse IP est connue, dans ce cas l’adresse Ethernet correspondante est mise à jour dans la table ARP de la machine courante.

Fin Tant que.

7) Lors d’un premier ping de A vers B, A envoi un ARP request à B pour connaître son adresse Ethernet. B répond. Cependant, B profite de cet ARP request pour enregistrer l’adresse Ethernet de A avec son adresse IP, économisant une requête ARP de sa part pour un éventuel envoi futur.

8) Un paquet ICMP est envoyé par A vers B sans paquet ARP request car A connaît déjà l’adresse de B dans sa table ARP suite au ping précédent. A la réception de ce paquet ICMP par B, un ARP request est envoyé de B vers A suivi d’un ARP reply de A vers B. En effet, B ne connaît plus A car il y a eu suppression. Ainsi, pour connaître l’adresse de destination du paquet ICMP reply, B doit envoyer un ARP request vers A.

9) C’est le timer de ping (1 seconde) qui re-déclenche l’émission d’un paquet ARP request. Le protocole ARP envoi juste un paquet pour demander l’adresse, si il n’a pas de réponse, il n’y a aucune réémission. Il est possible de tester cela avec l’envoi d’un seul paquet par le ping : ping –c 1. Nous constatons qu’il y a alors un seul paquet ARP envoyé. Pour info : timer ARP = 1200 secondes soit 20 minutes.

10) Lors de la configuration d’une interface, un paquet ARP gratuit est envoyé sur le réseau. Un paquet ARP gratuit est un paquet envoyé en broadcast (Ethernet) avec l’adresse IP de la machine ajoutée sur le réseau. Si une machine possède déjà cette adresse IP, elle va répondre à cet ARP gratuit request par un ARP gratuit reply pour annoncer que l’adresse IP associée à cette interface est déjà utilisée. Si il n’y a aucune réponse, pas de conflit IP.

Cet ARP gratuit permet alors de prévenir lorsqu’une même adresse IP est utilisée par deux machines différentes. Les deux machines concernées affiche le message d’alerte. De plus, si l’adresse IP est connue par une autre machine dans sa table ARP, l’adresse Ethernet correspondante est alors mise à jour lors de la réception de ce paquet ARP gratuit.

11)

14)

**Protocole ICMP :**

15)

16) Pour calculer les deux octets du checksum :

* Additionner par mot de 16 bits l’ensemble des données du paquet ICMP (sauf les deux octets du checksum)
* Réaliser le complément à 1 de ce résultat

Par exemple dans notre cas, voici les données de deux paquets ICMP :

1er paquet request : 0x 08 00 **60 8B** 12 32 85 42 00 00 00 … (suivi d’une multitude de zéro). Les octets 60 8B sont les octets du checksum. Il est donc necessaire d’additionner (0x) 08 00 + 12 32 + 85 42 (additionner les zéros suivants est bien évidemment inutile) = 0x 9F74. Puis le complément à 1 de 0x 9F74 = 0x 608B soit les octets du checksum.

2éme paquet reply : 0x 00 00 **68 8B** 12 32 85 42 … . Addition (0x) : 00 00 + 12 32 + 85 42 = 0x 9774. Complément à 1 = 688B. Correct !

**Protocole DHCP :**

17)

18)