LI310 - TME 2
Analyses de trames – outil WireShark

Benjamin BARON

1 Analyse manuelle de trames

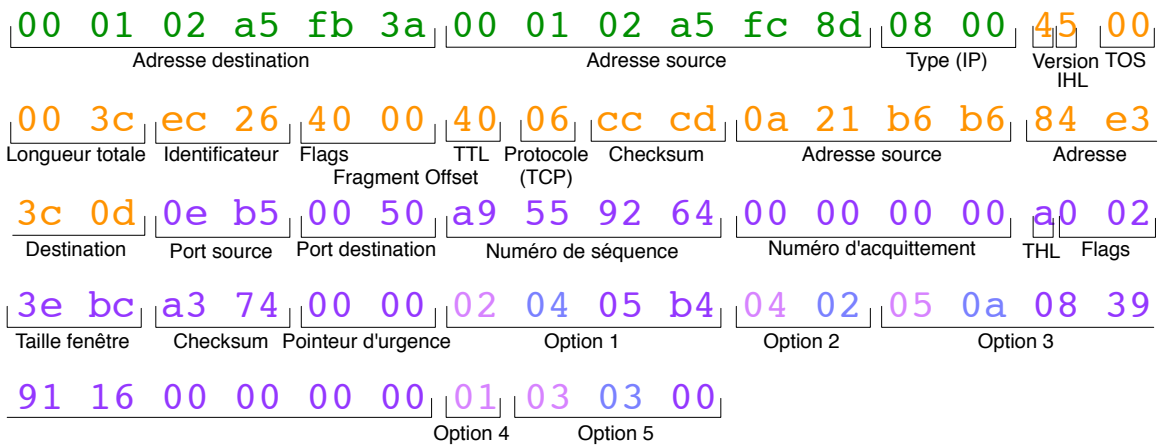

FIGURE 0.1: Trame ethernet décomposée

Question 1.1. Informations de niveau liaison :
- Adresse destination : `00:01:02:a5:fb:3a`
- Adresse source : `00:01:02:a5:fc:8d`
- Type de la trame : `0x0800` – DoD Internet (IP)

Question 1.2. Information de niveau réseau :
- `Version : 0x4` – Utilisation de IPv4
- `IHL : 0x5` – $5 \times 4 = 20$ octets $\Rightarrow$ délimiter la fin de l'entête IP
  Il n'y a donc pas d'options IP
- `ToS : 0x00`
- `Total Length : 0x003c` – Longueur paquet IP : $0x003c = 60$ octets $\Rightarrow$ délimiter du début à la fin du paquet (déterminer la présence ou non d'octets de bourrage dans Ethernet).
- `Identification : 0xec26` (mécanisme de fragmentation et réassemblage)
- `Flags` + champs `Fragment offset : 0x4000` :
  - Champ `Flag : 010` – `DF : 1` $\Rightarrow$ la fragmentation a été interdite par l'émetteur
  - Champ `Fragment offset : 0`
- `TTL : 0x40` – Le paquet est autorisé à traverser $0x40 = 64$ routeurs (valeur standard, donc paquet vient d'être envoyé)
- `Protocol : 0x06` – TCP
- `Checksum : 0xcccd` calculé uniquement sur l'entête
- Adresse IP de la source : `10.33.182.182`
- Adresse IP du destinataire : `132.227.60.13`

Question 1.3. Le paquet contient-il des options ?

On a `IHL : 0x5`, donc la longueur totale de l'entête est égale à 20 octets. Puisque c'est la taille minimale, on en déduit qu'il n'y a pas d'options dans ce paquet IP.

Question 1.4. Adresses IP du paquet :
- Adresse IP de la source : `10.33.182.182`
- Adresse IP du destinataire : `132.227.60.13`

Question 1.5. Protocole de transport utilisé : TCP du fait de la valeur `0x06` du champ `protocol` du paquet IP

Question 1.6. Informations de niveau transport :
- Port source : $0x0eb5 = 3765$
- Port destination : $0x0050 = 80$ – valeur associée à http. Ce segment est envoyé par un client à un serveur web
- Numéro de séquence (32 bits)
- Numéro d'acquittement (32 bits)
- `THL : 0xa` (ie. 10 mots de 4 octets) – entête de 40 octets ⇒ délimiter le segment TCP. La valeur minimale de `THL` est `0x5` ou 20 octets d'entête TCP
- `FLAGS TCP : 0x02` ⇒ `000010` (binaire). Seul `FLAG SYN : 1` ⇒ demande de connexion (`SYN`) envoyée à un serveur web par un client web
- `Window : 0x3ebc`. La taille de la fenêtre d'anticipation est de 16060 octets
- `Checksum` : calculé sur l'entête TCP + pseudo-header comprenant les adresses IP
- `Pointeur d'urgence : 0x0000` (si non nul, erreur)
- Partie options de TCP – suite d'options (une ou plusieurs) codées sur :
  - 1 octet : `00` ou `01`
  - Plusieurs octets – codage de type `TLV` – `Type` (1o) `Longueur` (1o) `Valeur` (octets suivants)

  Il s'agit d'un segment TCP contenant 5 options :
  - `0x02:04:05b4` – option codée sur plusieurs octets :
    - `type : 0x02` : négociation du MSS (*Maximum Segment Size*)
    - `longueur : 0x04` – 4 octets
    - `valeur : 0x05b4` – Longueur de MSS : $0x05b4 = 1560$ (ie. Le client ne peut pas recevoir des segments TCP contenant plus de 1560 octets)
  - `0x04:02` – option codée sur plusieurs octets :
    - `type : 0x04` – support des acquittement sélectifs (différent des acquittements positifs supportés à la base par TCP)
    - `longueur : 0x02` – 2 octets ⇒ il n'y a pas de valeur pour cette option
  - `0x08:0a:0839971600000000` – option codée sur plusieurs octets :
    - `type : 0x08` – estampille temporelle
    - `longueur : 0x0a` – 10 octets au total à partir de `0x08`
    - `valeur : 0x0839971600000000` – valeur de l'estampille temporelle
  - `01` – option `NOP` (no opertation)
  - `0x03:03:00` – option codée sur plusieurs octets :
    - `type : 0x03` – option associée à adaptation de la taille de la fenêtre
    - `longueur : 0x03` – 3 octets
    - `valeur : 0x00`

2 Utilisation de l'analyseur de trames WireShark

2.1 Introduction à WireShark

Question 2.1.1. Contenu des fenêtres de WireShark :
– Fenêtre du haut : liste des trames capturées et résume quelques caractéristiques de ces dernières.
– Fenêtre centrale : description précise de chaque champ de la trame sélectionnée.
– Fenêtre du bas : affichage de la trame en hexadécimal

Question 2.1.2. Format des données de la fenêtre du bas : Hexadécimal et ASCII

Question 2.1.3. Protocoles observés dans la capture : IP et ICMP

Question 2.1.4. Protocoles analysables par WireShark : ARP, DNS, TCP, HTTP, ICMP,...

2.2 Filtres d'affichage WireShark

Filtre qui ne sélectionne que les trames ARP de/vers l'interface ayant l'adresse MAC `00:1b:77:d2:d2:27`

```
arp.src.hw_mac == 00:1b:77:d2:d2:27 or arp.dst.hw_mac == 00:1b:77:d2:d2:27
```

3 Analyse d'une requête HTTP

Question 3.1. Protocoles présents : ARP, DNS, UDP, IP, TCP et HTTP

Question 3.2. Premiers échanges ARP :
– Objectif : résolution de l'adresse logique `ratp.fr`
– Premier paquet ARP : Obtenir l'adresse MAC de la machine ayant pour adresse IP `82.228.126.137` (ie. l'adresse IP de la passerelle du réseau local où se trouve la machine émettrice du paquet ARP)
  – Source : `82.228.126.137`
  – Destinataire : brodacast – `82.228.126.00` (ie. Tout le réseau local)
– Second paquet : L'adresse MAC cherchée a été trouvée et adressée à l'émetteur du premier paquet.
  – Adresse MAC obtenue (source) : `3a:b7:9b:5f:47:cc`
  – Destinataire : `00:1b:77:d2:d2:27`

```
Sender MAC address: 00:1b:77:d2:d2:27 (00:1b:77:d2:d2:27)
Sender IP address: 82.228.126.137 (82.228.126.137)
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
Target IP address: 82.228.126.254 (82.228.126.254)
```

Question 3.3. Objectif des trames 3 à 6 : Echange de trames entre le client et un serveur de DNS :
– Adresse IP du client : `82.228.126.137`
– Adresse IP du serveur de DNS : `212.27.53.252`
Le client demande l'adresse IP du site `ratp.fr` – réponses du serveur de DNS :

```
ratp.fr: type A, class IN, addr 81.255.174.189
ratp.fr: type A, class IN, addr 62.160.111.1
```

Il y a donc deux adresses IP correspondant au site `ratp.fr`
– Trames 3 à 4 : `Standard query A ratp.fr` – requête de l'adresse IPv4 de `ratp.fr`
– Trames 5 à 6 : `Standard query AAAA ratp.fr` – requête de l'adresse IPv6 de `ratp.fr`

Question 3.4. Protocole de transport associé au protocole applicatif utilisé dans les trames 3 à 6.
– Protocole de transport : UDP (*User Datagram Protocol*)
– Protocole applicatif : DNS (*Domain Name System*)
Fonctionnalités d'UDP : communication entre processus de machines sur un même réseau – multiplexage, détection d'erreurs, mode non connecté et plus rapide que le TCP

Question 3.5. But de l'échange : Chargement de la page `http://ratp.fr`

Question 3.6. Plusieurs requêtes GET : lors du chargement d'images extérieures au site `ratp.fr` ou javascript présents sur la page chargée.

Question 3.7. A partir de la trame 7 : utilisation du protocole TCP

Fonctionnalités de TCP : fiable, contrôle de flux, contrôle d'erreurs, contrôle de congestion.

Trames 3 à 6 : utilisation du protocole UDP pour les requêtes DNS.

Question 3.8. Filtre pour sélectionner les trames relatives au protocole http : `tcp.port == 80`

Question 3.9. Trame 45 : contenu d'une page html du site `ratp.fr`. Elle correspond à la fin de l'envoi du fichier HTML correspondant à la page web. Le fichier HTML a été envoyé dans 3 segments séparés – les trames 40, 41 et 45. Il s'agit de paquets TCP fragmentés (*Reassembled TCP segments*)

Question 3.10. Seconde requête DNS : obtenir l'adresse IP correspondant à l'adresse logique `logc5.xiti.com` – code javascript qui force l'appel à une autre page.

4 Analyse des commandes echo request (PING) et echo reply d'ICMP

Question 4.1. Filtre `ip.proto == 1` : affiche les trames contenant des paquets IP

Question 4.2. `ping -c 4 ufr-info-p6.jussieu.fr`

Envoi de 4 paquet `echo request` à l'adresse `ufr-info-p6.jussieu.fr` (champ type du paquet ICMP = 8)

4.3 Première trame ICMP – *echo request*

Question 4.3.1. Adresses IP source :
– Adresse IP de la machine source : `132.227.110.115`
– Adresse de sous-réseau hébergeant la machine source : `132.227.110.0`
– Masque de sous-réseau : `255.255.255.0`

Question 4.3.2. Adresses IP distantes :
– Adresse IP de la machine distante : `132.227.68.44`
– Adresse de sous-réseau hébergeant la machine distante : `132.227.68.0`
– Masque de sous-réseau : `255.255.255.0`

Remarque. Les deux machines ne se trouvent pas dans le même sous-réseau puisque le subnetid est différent.

Question 4.3.3. Adresse MAC source de la trame : `00:0d:5e:dc:39:74`. Cette adresse correspond à la machine émettrice de la trame contenant le paquet ICMP `echo` d'adresse IP `132.227.110.115`

Question 4.3.4. Adresse MAC destination de la trame : `00:00:5e:00:01:6e`. Cette adresse correspond au routeur du réseau local où se situe la machine ayant envoyé la trame `echo` – adresse de la passerelle

Question 4.3.5. Datagramme IP non fragmenté : `Flags : 0x04` (Don't Fragment) – flag DF : 1

Question 4.3.6. Longueur de l'entête IP : 20 octets, donc puisque c'est la taille minimale d'une entête IP, alors ce paquet ne contient pas d'options.

Question 4.3.7. Longueur du champ de données (`data`) du datagramme IP : 64 octets (84 TL - 20 IHL)

Question 4.3.8. Champs de l'entête ICMP – 8 octets :
- `Type : 0x08` – `echo request` (`ping`)
- `code : 0x00`
- `checksum : 0xf3b6`
- `identifier : 0xa76a`
- `sequence number : 0x0000`

Question 4.3.9. Longueur des données ICMP : 56 octets Par défaut, la commande `ping` envoie 56 octets (spécifié dans l'option `-s` de la commande `ping`)

Remarque. Les octets du champ data sont "bidons".

4.4 Seconde trame ICMP – *echo reply*

Question 4.4.1. Longueur de l'entête IP : 20 octets ⇒ pas de champ `option`

Question 4.4.2. Longueur du champ data du datagramme IP : 64 octets = 84 TL - 20 IHL

Question 4.4.3. Champs de l'entête ICMP – 8 octets :
- `type : 0x00` – `echo reply` (`ping`)
- `code : 0x00`
- `checksum : 0xfbb6`
- `identifier : 0x7a76a` – même identifiant que pour le paquet `echo request`
- `sequence number : 0x0000`

Type différent, même identifiant puisqu'il s'ait de la réponse à la requête `echo`.

Question 4.4.4. Données du message ICMP :

```
                                    cb 3c 33 47 6f 57
0030  04 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35
0060  36 37
```

Données du message précédent :

```
                                    cb 3c 33 47 6f 57
0030  04 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35
0060  36 37
```

On constate que ce sont les mêmes données.

Question 4.5. Entêtes IP des trames `echo request` :
- Version : 4

– Header length : 20 bytes
– Differentiated Services Field : `0x00` (DSCP `0x00` : Default ; ECN : `0x00`)
– Total Length : 84
– Identification : `0x0000` (0)
– Flags : `0x04` (Don't Fragment)
– Fragment offset : 0
– Time to live : 64
– Protocol : ICMP (`0x01`)
– Header checksum : `0x7e43` [correct]
– Source : `132.227.110.115` (`132.227.110.115`)
– Destination : `132.227.68.44` (`132.227.68.44`)
Seul les champs `checksum` et `identification` sont différents par rapport au premier paquet émis. En effet, le champ identification est incrémenté de 1 à chaque `echo request` émis ; de ce fait, puisque le champs `identification` est différent, alors le champ `checksum` est différent.

Entête ICMP des trames `echo request` :
– Type : 8 (`Echo (ping) request`)
– Code : 0 ()
– Checksum : `0x0fb2` [correct]
– Identifier : `0xa76a`
– Sequence number : 0 (`0x0000`)
De même, les champs `checksum` et `sequence number` sont différents. Le champ `sequence number` est incrémenté de 1 à chaque `echo request` émis.

Commande `ping -c 1 -s 2000 www.lip6.fr`

Remarque. Taille max d'une trame Ethernet : 1500 octets

4.6 Première trame de l'échange

Question 4.6.1. Entête IP : `Flags: 0x02 (More Fragments)` indique que le datagramme a été fragmenté – bit `MF = 1`

Question 4.6.2. Dans l'entête IP, le champ `Fragment Offset: 0` indique que le datagramme est le premier fragment.

Question 4.6.3. Longueur totale du datagramme indiquée par le champ `Total Length: 1500` du datagramme. Longueur du champ data du datagramme : 1480 = 1500 TL - 20 IHL

Question 4.6.4. Adresse de la machine distante `www.lip6.fr` : `132.227.73.20`

4.7 Deuxième trame de l'échange

Question 4.7.1. Champ `Fragment offset: 1480` indique que ce paquet n'est pas le premier envoyé.

Question 4.7.2. Champ `Flags: 0x00` donc c'est le dernier fragment – bit `MF = 0`

Question 4.7.3. Longueur totale de ce datagramme identifié par le champ `Total length : 548`

Question 4.7.4. Longueur de ces deux fragments : $1500 + 548 = 2048$ octets

Question 4.7.5. Les deux paquets ont été envoyés avec 20 octets d'entête.
– D'une part, on a $1548 - 40 = 1508$ octets de données envoyées.
– D'autre part, on a $548 - 20 = 528$ octets de données envoyées.

$\Rightarrow 1480 + 528 = 2008$ octets de données IP dont 8 octets d'entête ICMP.Ainsi, il y a 2000 octets de `data` ICMP.

Question 4.8. Le paquet ICMP `echo reply` a été fragmenté en deux fragments, comme pour le paquet ICMP `echo request`.

Question 4.9. Commande `ping -c 1 -s ? rp.lip6.fr`

Le paquet ICMP `echo request` a été fragmenté en 3, la longueur totale du fragment est 3068 octets :

$$1500 - 20 + 1500 - 20 + 68 - 20 - 8 = 3000$$

La commande tapée est donc : `ping -c 1 -s 3000 rp.lip6.fr`