UFR Math-Info
Master 2 SIRIS

Sécurité Avancée
CI 1

# Scan

## I) Objectifs

1. comprendre les filtres pcap pour capturer des trames,
2. prendre en main un outil de scan,
3. apprendre à détecter différents types de scan,
4. concevoir une détection automatisée des scans.

## II) Matériel

NB : c'est une étape optionnelle ; seul tcpdump, wireshark et un shell seront nécessaires pour faire le TP. Si votre machine a déjà tout cela, il n'est pas utile de télécharger la VM.

Télécharger la VM : https://seafile.unistra.fr/f/c0fd6bd6e9b64a37b0b3/?dl=1

## III) Analyse de trame (rappels)

Cette première partie va vous faire manipuler des filtres pcap.

Ces filtres serviront ensuite pour faire du forensique réseau (analyse de paquets réseau dans le contexte de la sécurité informatique).

### a) Contexte d'utilisation

Souvent, un dysfonctionnement applicatif, un problème de configuration ou d'architecture réseau, une charge importante, ou un incident de sécurité sont difficilement explicables si on ne dispose pas d'informations précises. Dans tous ces cas, l'analyse de trame peut apporter les éléments nécessaires. L'analyse de trame est à l'administration réseau ce que le mode trace du debugger est à la programmation. Analyser des trames peut aussi aider à comprendre comment les protocoles fonctionnent. (cf. les ouvrages de Richard Stevens).

### b) Présentation de tcpdump et libpcap

Le plus connu des logiciels graphique d'analyse de trame est Wireshark.

Mais nous allons utiliser tcpdump pour commencer. C'est un outil qu'il est important de matriser, et ceci pour plusieurs raisons :

1. tcpdump est un outil en ligne de commande plus facile à utiliser à distance (ssh),
2. on peut faire des scripts (par exemple, avec l'option -c N pour qu'il s'arrête après avoir capturé N paquets, ou bien les options -w/-r pour traiter des fichiers),
3. les filtres de capture ont la même syntaxe dans wireshark, tshark et tcpdump (les *display filters* ont malheureusement une autre syntaxe).
4. la bibliothèque sur laquelle s'appuie tcpdump, libpcap est universelle, et sert de base à de nombreux autres outils.

## c) Filtres pcap

Lorsque l'on démarre tcpdump, des nombreuses trames s'affichent (en fonction de l'activité réseau). C'est un phénomène courant en administration réseau et en sécurité : la quantité d'information est trop importante.

C'est pourquoi on va mettre en place des filtres pour sélectionner précisément le trafic que l'on veut observer.

Pour filtrer les paquets, il est possible d'agir au niveau de différentes couches protocolaires : liaison (Ethernet), réseau (IP), transport (TCP), application...

Dans tcpdump, la mise en place de ces filtres se base sur un langage simple.

NB : La page de manuel de *pcap-filter* détaille la syntaxe des filtres.

### c).1 Syntaxe de base

La syntaxe générale de tcpdump prend les formes suivantes :

```
tcpdump protocole
```

Avec protocole :

- ether,
- ip,
- arp,
- tcp,
- udp,
- icmp
- etc.

```
tcpdump protocole type identifiant
tcpdump protocole direction identifiant
```

avec

- protocole : ether, ip, arp, tcp, udp
- type : host, port
- direction : src, dst
- identifiant : adresse, numéro de port

### c).2 Combinaison de filtres

Il est possible de combiner ces expressions avec des opérateurs logiques "and", "or" et "not".

NB : Attention à la syntaxe de la commande tcpdump : il faut mettre d'abord toutes les options, puis le filtre pcap. Il est fortement recommande de mettre le filtre entre guillements. Exemples :

```
tcpdump -n -s 1500 -i eth0 "ip host 130.79.14.177 and udp dst 123"
tcpdump -r a.pcap -w out.pcap "ip host 192.0.9.1"
```

### d) Exercice

Expliquer à quoi s'appliquent les filtres suivants :

1. tcpdump arp
2. tcpdump icmp
3. tcpdump ip
4. tcpdump 'ether host 00:10:20:41:5A:F0'
5. tcpdump 'ip src 192.168.56.1'
6. tcpdump 'ip dst 192.168.56.200'
7. tcpdump 'tcp port 22'
8. tcpdump 'udp port domain or tcp port http'

### e) Filtres avancés

Lorsque les éléments de syntaxe de base ne suffisent plus, il est possible de filtrer des trames en précisant :

1. l'offset, c'est à dire la position de l'octet dans la trame,
2. et la longueur de l'en-tête que l'on veut tester.

sous la forme suivante :

```
protocol [16:2]
```

[16 :2] signifie prendre 2 octets à partir du 16$^{ème}$ octets de la trame
Exemples :

- tcpdump 'ip[9] $>$ 200' : valeur du 9$^{ème}$ octet de l'en-tête ip $>$ 200 .
- tcpdump 'ether[0 :3] = 0x00000c' : les 3 premiers octets d'une trame ethernet qui commencent par 0000c0 en hexadecimal, un O.U.I. de Cisco
  Attention, ether[0:3] est *exemple fictif ; il ne fonctionne pas ; en raison des contraintes d'alignement des octets, la longueur doit être égale à 1, 2 ou 4)*

Remarque : Il est important de connaître les options -n et -s de tcpdump. Elles servent (respectivement) :

- -n pour ne pas faire la résolution DNS inverse (adresse IP  nom de domaine)
- -s N pour préciser la taille des trames capturées afin de prendre les trames complètes (sur certaines implémentations, seuls les premiers octets sont capturés pour des raisons de performance).

Plusieurs opérations arithmétiques et logiques sont possibles sur les valeurs désignées par la syntaxe précédente (protocole[offset:taille]) :

```
&, | , =, != ,<, > ...
```

Ces opérations seront utilisées pour faire des masques afin d'isoler certains bits, comme dans l'expression suivante :

```
mac[0]&0x40 = 0x00
```

## f) Comment fonctionnent les filtres ?

Les filtres donnés à tcpdump sont compilés dans du bytecode qui est interprété par une pseudo-machine qui va matcher les paquets. La fonction du noyau qui réalise ce filtrage s'appelle BPF (Berkeley Packet Filter). BPF dispose d'un registre, d'un accumulateur et de quelques instructions (charge un emplacement mémoire dans l'accumulateur, comparer à une valeur, faire un saut etc.)

Pour voir le code généré, dans un syntaxe similaire à un assembleur, utiliser l'option *-d* de tcpdump. Exemple :

```
# tcpdump -d tcp port 22
(000) ldh [12] ; load ether type
(001) jeq #0x86dd jt 2 jf 8 ; IPv6 ?
(002) ldb [20] ; load v6 Next Header field
(003) jeq #0x6 jt 4 jf 19 ; TCP ?
(004) ldh [54] ; load source port
(005) jeq #0x16 jt 18 jf 6 ; Source port 22 ?
(006) ldh [56] ; load dest. port
(007) jeq #0x16 jt 18 jf 19 ; Dest. port 22 ?
(008) jeq #0x800 jt 9 jf 19 ; IPv4 ?
(009) ldb [23] ; load proto
(010) jeq #0x6 jt 11 jf 19 ; TCP ?
(011) ldh [20] ; load ip fragment id field
(012) jset #0x1fff jt 19 jf 13 ; mask (keep frags)
(013) ldxb 4*([14]&0xf) ; compute ip header size
(014) ldh [x + 14] ; load source port
(015) jeq #0x16 jt 18 jf 16 ; Source port 22 ?
(016) ldh [x + 16] ; load dest. port
(017) jeq #0x16 jt 18 jf 19 ; Dest. port 22 ?
(018) ret #262144    ; return ok
(019) ret #0    ; return error
```

cf. https://www.kernel.org/doc/Documentation/networking/filter.txt

## g) Exercices

Écrire les filtres suivants :

1. Afficher les paquets d'un ping (requête et réponse) en visant précisément les éléments de protocole impliqués (c'est-à-dire les octets et les valeurs correspondants).
2. Isoler les datagrammes UDP dont le port destination est compris entre 33434 et 33534 ainsi que les trames ICMP *port unreachable* et *ttl expired*. Comment peut-on tester simplement si le filtre fonctionne ? A quoi correspond ce trafic ?
3. Voir les connexions HTTP initiées vers 193.0.6.139 (connexion = uniquement le premier paquet TCP)(indice : utiliser les offset et les masques)
4. Afficher toutes les trames Ethernet issues d'une interface réseau dont l'O.U.I. (Organizationally Unique Identifier) est donné au tableau et à destination de l'adresse de diffusion

# IV) Scan

Un scan est une opération de reconnaissance qui permet de déterminer quels sont les ordinateurs et services existants, très souvent dans le but de les compromettre. Mais le scan a aussi des usages légitimes : cartographie, audit etc.

Il existe plusieurs types de scan :

- scan de différentes adresses IP d'un réseau pour identifier les machines existantes (scan horizontal)
- scan d'une adresse IP sur différents ports pour identifier les services accessibles (scan vertical)

Le scan est une pratique extrêmement répandue sur Internet, à tel point qu'elle constitue un bruit de fond permanent. Un machine piratée contrôlée par un cybercriminel possède une valeur d'usage : elle lui permet d'attaquer (scan, deni de service), de stocker des données, d'anonymiser les accès, etc.

L'enjeu est donc d'avoir le plus possible de machines sous son contrôle.

Pour ce faire, les cybercriminels s'appuient sur des scans automatisés. Ils sont émis par un ensemble de machines piratées fonctionnant en parallèle, les *botnets*.

En raison de la fréquence importante de ces scans automatisés, une machine vulnérable accessible sur Internet est compromise en l'espace de quelques minutes.

Sans faire de scan à l'échelle industrielle, nous allons dans la suite de l'énoncé utiliser *nmap* pour scanner une machine et découvrir les techniques de base.

## a) Premier contact

Installer nmap sur la machine et parcourir la documentation pour comprendre comment il s'utilise. Faire quelques essais de scans. NB : pour limiter l'impact, faire des scans de votre propre adresse IP ou de l'adresse de la machine hôte (si vous utiliser une VM).

## b) Types de scan

Suivre le tutoriel https://nmap.org/bennieston-tutorial/ pour voir les différentes methodes de scan.
Mettre en place une capture et observer le trafic produit pour scanner *un seul port*
Essayer les options -sT, -sS, -sA, -sF, -sN, -sX, -P0
NB : la cible du scan doit être votre propre machine.
Instructions :

1. Lancer un capture sur l'interface de loopback (le filtre ignore le trafic DNS et ne prend en compte que l'IPv4, la commande utilisée n'est pas tcpdump mais tshark car les paquets TCP SYN sont plus lisibles) :

   ```
   tshark -i lo -f '(not udp port 53) and ip'
   ```

2. Pour lancer un scan de type "-sT" sur le port SSH sans ping préalable (NB : tester avec un port censé être ouvert, 22 par exemple, et un port fermé, 23 par exemple, et noter les différences de comportement.)

   ```
   nmap -P0 -sT -p 22 127.0.0.1
   ```

### c) Scan de plusieurs port

Lancer nmap pour scanner les ports ouverts/fermés de votre ordinateur et capturer les paquets. Quels sont les caractéristiques du trafic correspondant à ce scan ?

## V) Forensique réseau

Telecharger le fichier pcap : https://seafile.unistra.fr/f/cf2b721d4ffd4828a197/?dl=1

1. Vérifier que la signature md5 du fichier correspond bien à 320501caa4c94f6aac5a1805050e03e3
2. Quelle est l'adresse IP de l'attaquant ?
3. Quelle est l'adresse IP destination ?
4. Plusieurs méthodes de scan différentes ont été utilisée. Quelles sont-elles ? Expliquer comment chacune d'elle fonctionne.
5. Quel outil de scan a été utilise ? Comment avez-vous fait pour le déterminer ?
6. Quel est le but du scan de port ?
7. Quels sont les ports qui sont ouverts sur la destination ?
8. Quel système d'exploitation a été utilise ?

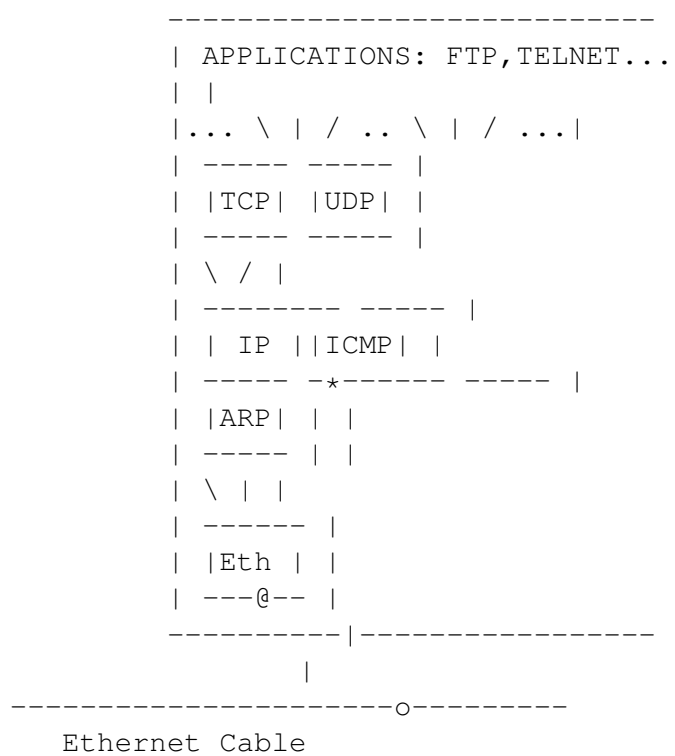
## VI) Détection de scan

1. Comment détecter qu'un scan est en cours sur une machine ?
2. Quelle structure de données utiliser ?
3. Proposer un algorithme avec un seuil de détection.
4. Implémenter cet algorithme dans le langage de votre choix.
5. Votre programme doit pouvoir lire un fichier pcap. Tester le fichier pcap du point précédent sur votre programme.

# VII) Annexe : structure des trames

## a) Architecture des protocoles TCP/IP

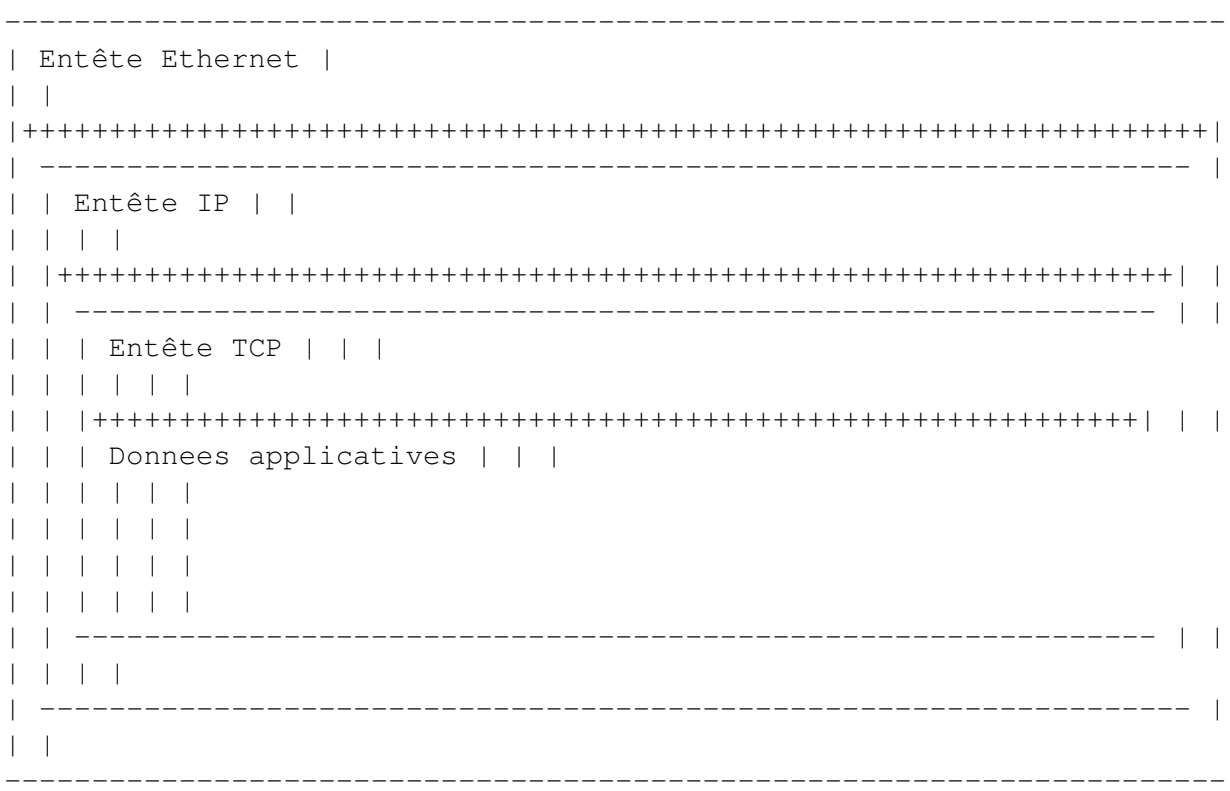
(selon la rfc1180)



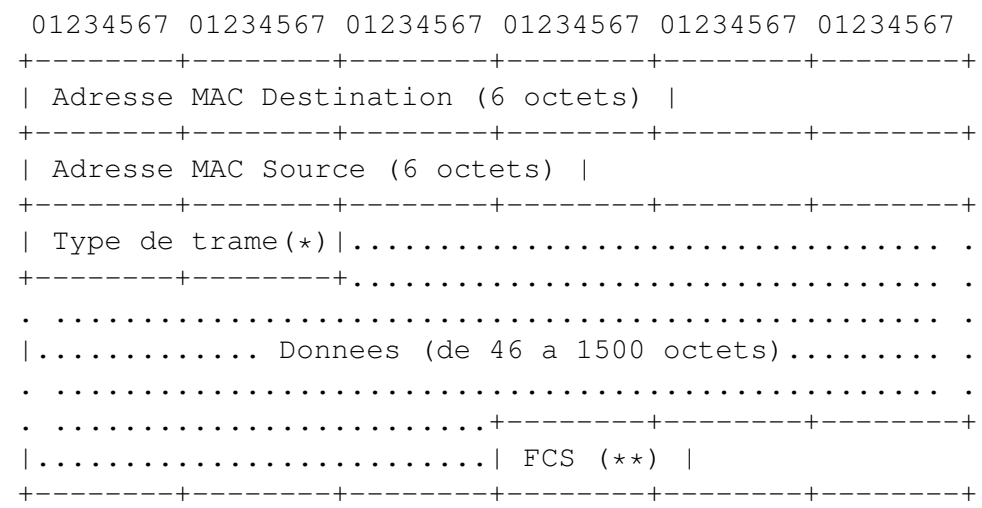

IP : Internet Protocol
UDP : User Datagram Protocol
TCP : Transmission Control Protocol

ICMP : Internet Control Message Protocol
ARP : Address Resolution Protocol
Applications : ftp, http, smtp, snmp, nfs,

### b) Exemple : structure d'une trame TCP sur un LAN



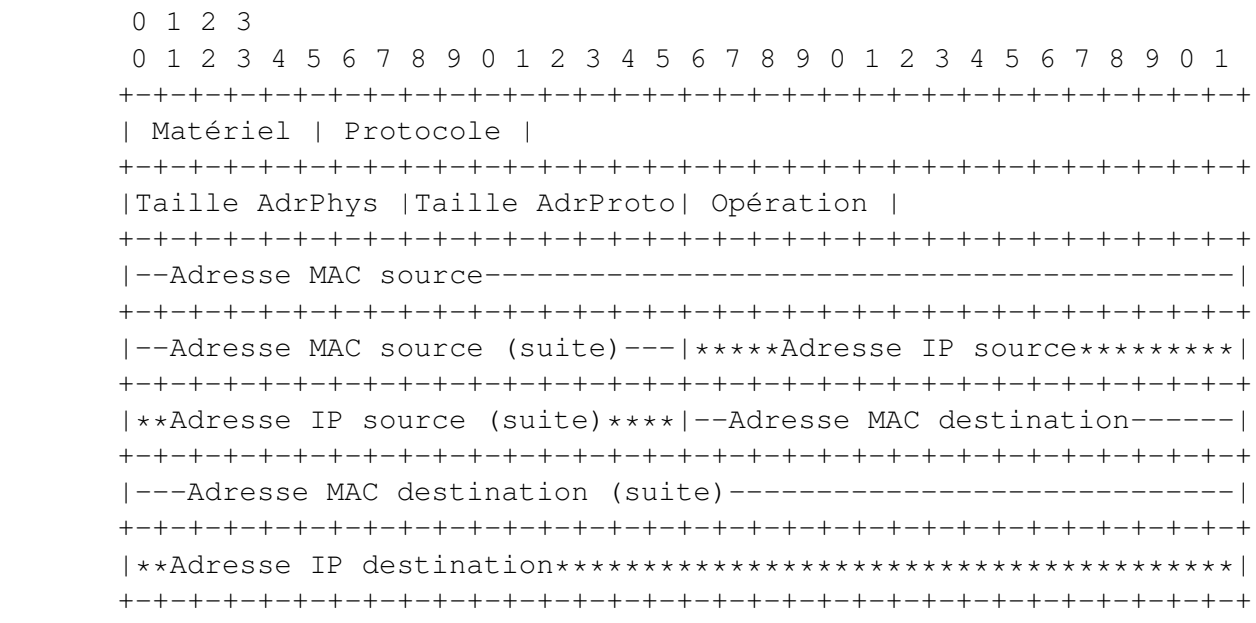
### c) Format d'une trame Ethernet




(*) Valeur pour le type de trame :
(hexadecimal)
0800 : IP (Internet protocol)
0806 : ARP (Address Resolution Protocol)
8035 : RARP (Reverse Address Resolution Protocol)

(**) FCS : Frame Check Sequence – Somme de contrôle de la trame
valeur calculée par la carte Ethernet

### d) Format d'un paquet ARP (RFC 826)

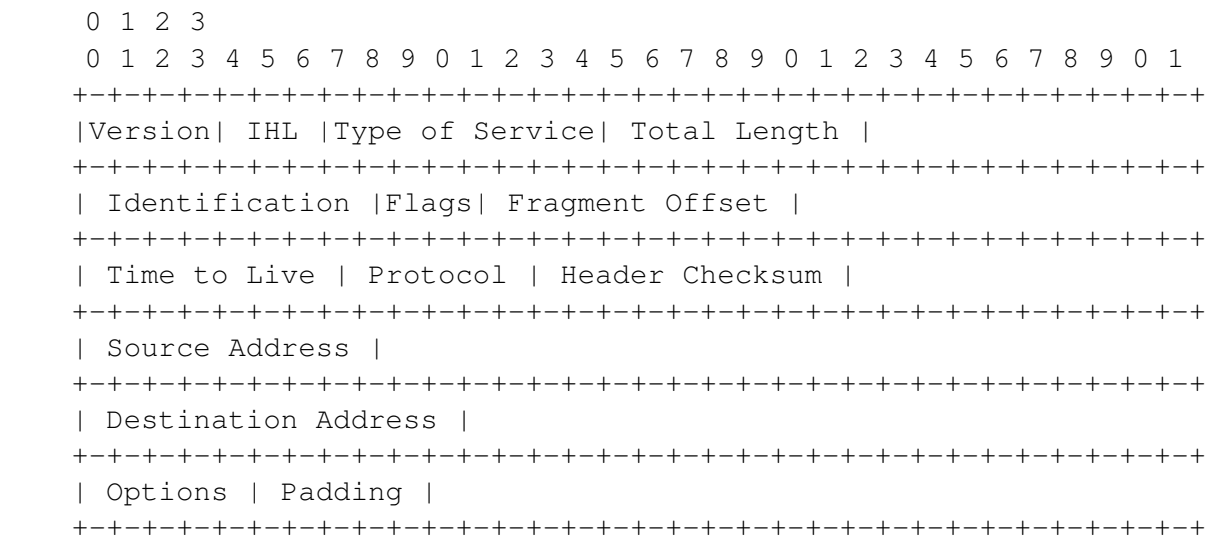



Remarque : "Type de trame" dans l'entête ethernet : 0806
(la trame ethernet n'est pas représentée ici)

Matériel : 1 pour ethernet
Protocole : 0800 pour IP
Taille AdrPhys : Taille de l'adresse physique, normalement 6
car une adresse MAC a 6 octets de long
Taille AdrProto : Taille de l'adresse protocole, normalement 4,
car une adresse IP fait 4 octets

Opération : prend les valeurs suivantes :
1 Requête ARP
2 Réponse ARP

### e) Format de l'entête IP (RFC 791)




Version: (4 bits) Normalement égal à 4

IHL: (4 bits) Internet Header Length. Longueur de l'entête en nombre de mots de 32 bits; au minimum égal à 5. La taille de l'entête est variable à cause des options.

Type of Service: (1 octet) Qualité de service ; rarement implementé

Total Length: (2 octets) Longueur totale du datagramme IP en octets, qui inclus l'entête et les données. Si il y a plusieurs fragments, cette taille ne concerne que le fragment courant.

Identification: (2 octets) Une valeur permettant de réassembler les fragments s'il y a lieu.

Flags: (3 bits)

```
Various Control Flags.
  Bit 0: reserved, must be zero
  Bit 1: 0 = May Fragment, 1 = Don't Fragment.
  Bit 2: 0 = Last Fragment, 1 = More Fragments.
```

Fragment Offset: (13 bits) Position du fragment. Mesuré en bloc de 8 octets. Le premier fragment a un offset à 0.

Time to Live: (1 octet) Indique le temps maximum qu'un datagramme IP est autorisé à rester sur le réseau. Il est décrémenté à chaque passage dans

un routeur. Lorsqu'il vaut 0, le datagramme est détruit.

Protocol: (1 octet) Type de protocole transporté :
6 pour TCP, 17 pour UDP, 1 pour ICMP

Header Checksum: (2 octets) Somme de contrôle calculée sur l'entête.

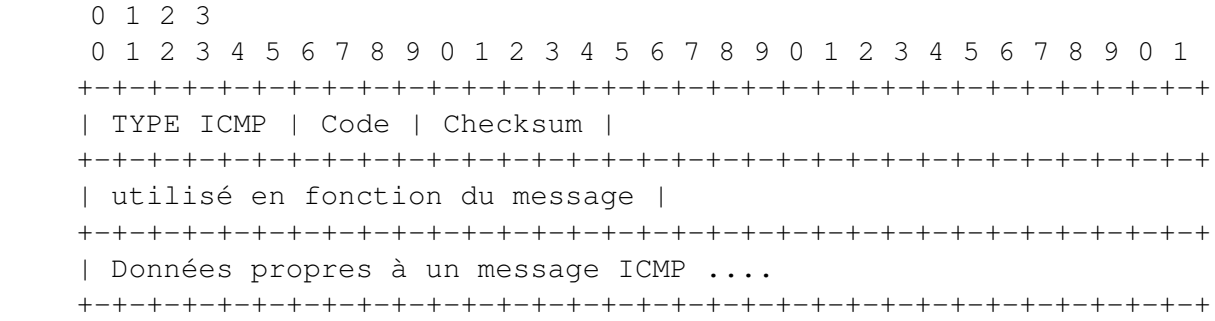
Source Address: (4 octets) Adresse IP source

Destination Address: (4 octets) Adresse IP destination

Options : [pas intéressant ici ; documenté dans la rfc 791]

## f) Format d'un paquet ICMP (RFC 792)

Remarque : un paquet ICMP possède une entête IP



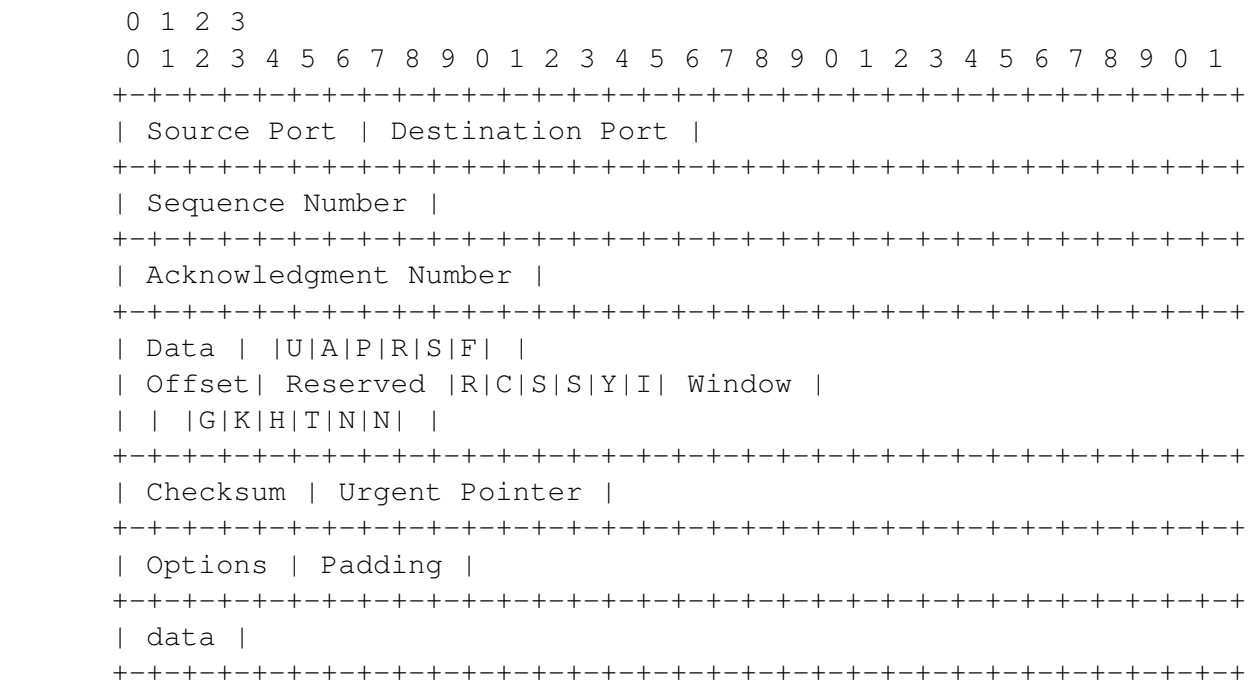

Dans l'entête IP :
Version: (4 bits) Normalement égal à 4
IHL: (4 bits) Internet Header Length. (cf IP)
Type of Service: (1 octet) égal 0
Protocol: (1 octet) Type de protocole transporté : égal à 1 pour ICMP
Source Address: (4 octets) Adresse IP source du message ICMP
Destination Address: (4 octets) Adresse IP destination du message ICMP

TYPE ICMP :

8 : echo = PING
0 : echo reply = Réponse au ping
3 : Destination Unreachable
11 : Time Exceeded
12 : Parameter Problem Message
4 : Source Quench Message
5 : Redirect Message
13 : timestamp message;
14 : timestamp reply message.

Code : dans certains cas, précise la raison

## g) Format de l'entête TCP (RFC 793)




Source Port: (2 octets)
Destination Port: (2 octets)

Sequence Number: (4 octets) Numéro de séquence du premier octet de données

Acknowledgment Number: (4 octets) Numéro de séquence du premier octet de données attendu)

Data Offset: (4 bits) Taille de l'entête en nombre de mot de 4 octets.

Reserved: (6 bits) Réservé ; doit être à 0

Control Bits: (6 bits) (de gauche à droite)

URG: Urgent Pointer
ACK: Acknowledgment
PSH: Push Function
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: No more data from sender

Window: (2 octets) taille de la fenêtre ; nombre d'octets de données (à partir du numéro de séquence indiqué dans "Acknowledgment Number") que l'emetteur

est disposé à accepter.

The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

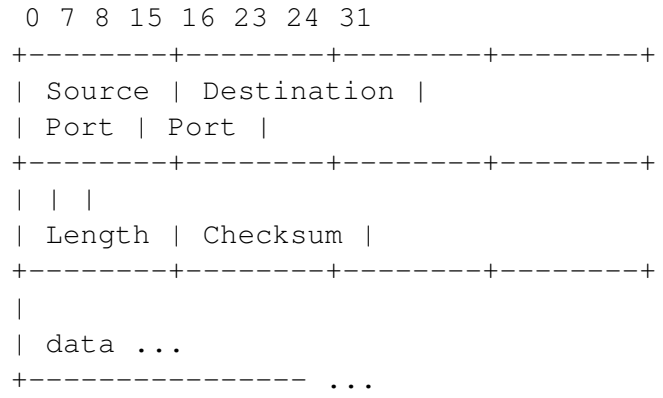
Checksum: (2 octets) Somme de contrôle. Permet de vérifier l'intégrité des données.

Urgent Pointer: (2 octets) [pas intéressant ; documenté dans la rfc793]

Options: [pas intéressant]

Padding: bourrage ; pour s'assurer que l'entête s'arrête bien sur un multiple de 32 octets.

### h) Format de l'entête UDP (RFC 768)




Length : taille de l'entête et des données
Checksum: (2 octets) Somme de contrôle. Permet de vérifier l'intégrité des données.