R2.05 – Les protocoles reseaux

RESPONSABLE : CRISTINA ONETE

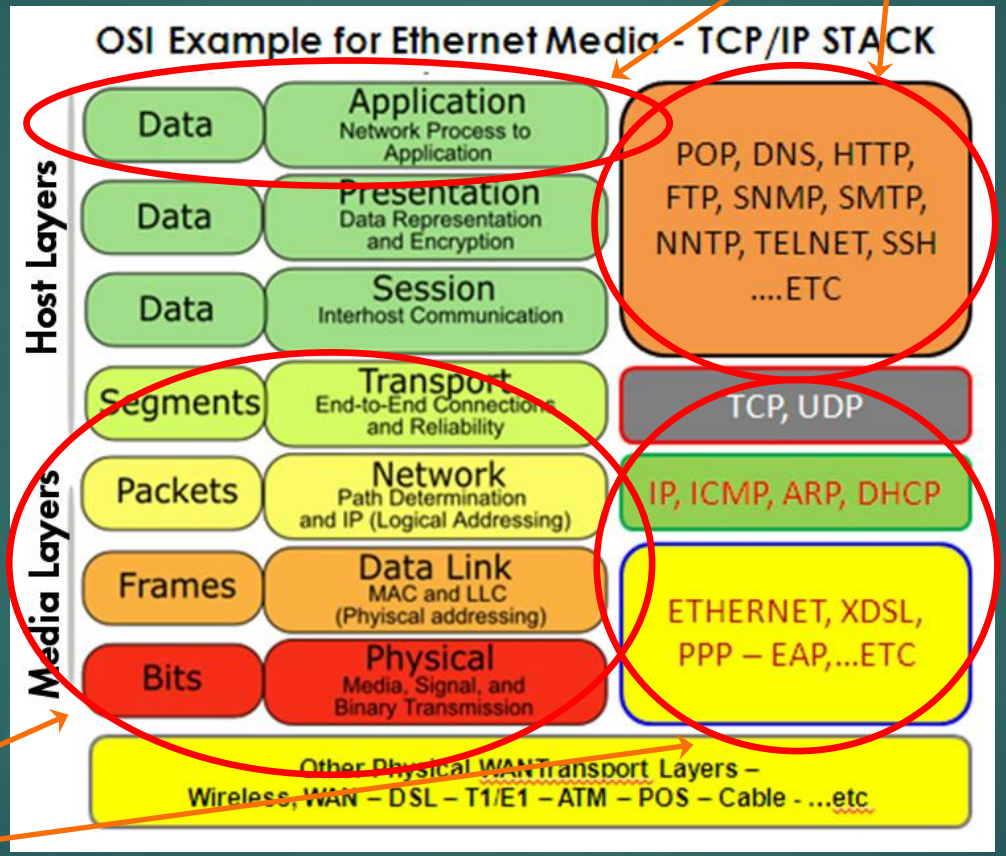
MATERIEL : HTTPS://WWW.ONETE.NET/TEACHING.HTML

EMAIL : CRISTINA.ONETE@GMAIL.COM

R2.04, R2.05

R2.05

Les couches architecturales des réseaux



Les protocoles des réseaux

R2.04

Source : ipv6.com

Protocoles couche 3 : ICMP, DHCP

ICMP & les transmissions fiables

- Problématique :
  - Protocole IP : assure la transmission de messages entre deux sous-réseaux
  - Mais les messages IP ne sont pas échangés de façon fiable
  - Comment détecter les erreurs ?

- Le protocole ICMP est utilisé pour gérer des erreurs :
  - Destination non trouvée/ non disponible/ n'existe pas
  - Duree de vie dépassée
  - Message mal-formé
  - Reconnaissance de topologie : ping
  - ... etc.

Format messages ICMP

Bit 0 - 7	Bit 8 - 15	Bit 16 - 23	Bit 24 - 31
Type de message	Code	Somme de contrôle	
Bourrage ou données			
Données (*optionnel et de longueur variable*)			

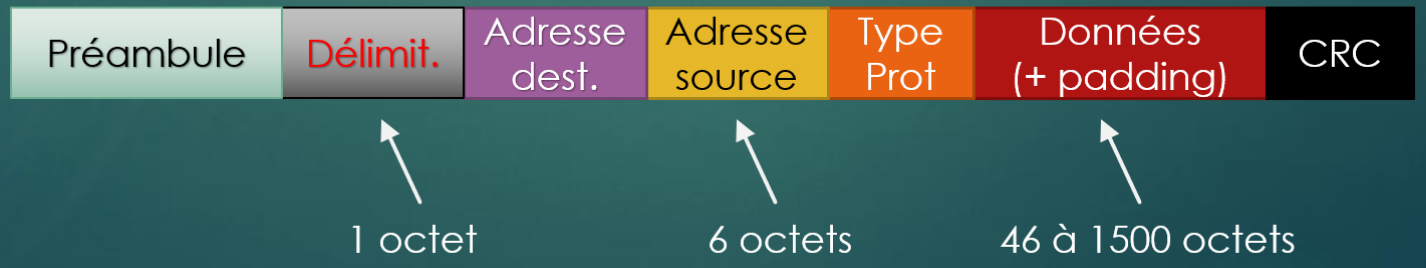
- Pour chaque type de message, plusieurs codes possibles :
  - Type : type de message
  - Code : détails sur le type de messages
- Exemples :
  - Type 3 : Destination Unknown, code 6/7 : réseau/machine destination inconnue
  - Type 11 : Time exceeded, code 0 : TTL expiré
  - Type 12 : Bad IP header, code 1: paramètre manquant; code 2 : mauvaise taille

Encapsulation

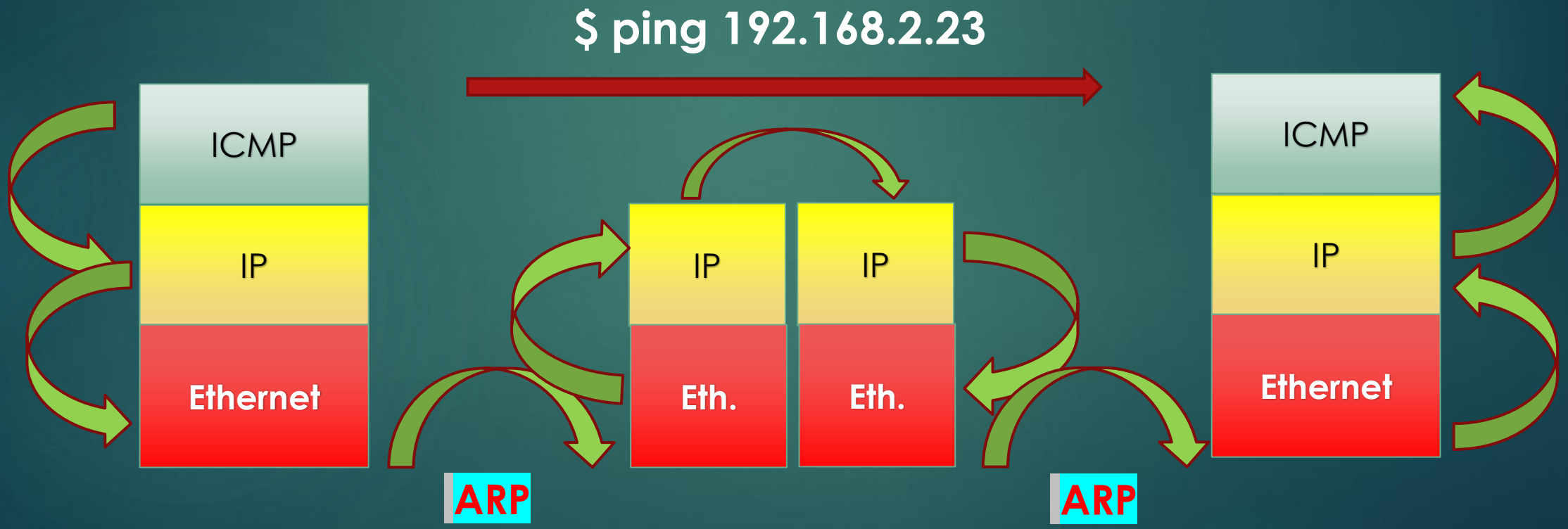
Bit 0 - 7	Bit 8 - 15	Bit 16 - 23	Bit 24 - 31
Type de message	Code	Somme de contrôle	
Bourrage ou données			
Données (*optionnel et de longueur variable*)			

<------ 32 bits ------>

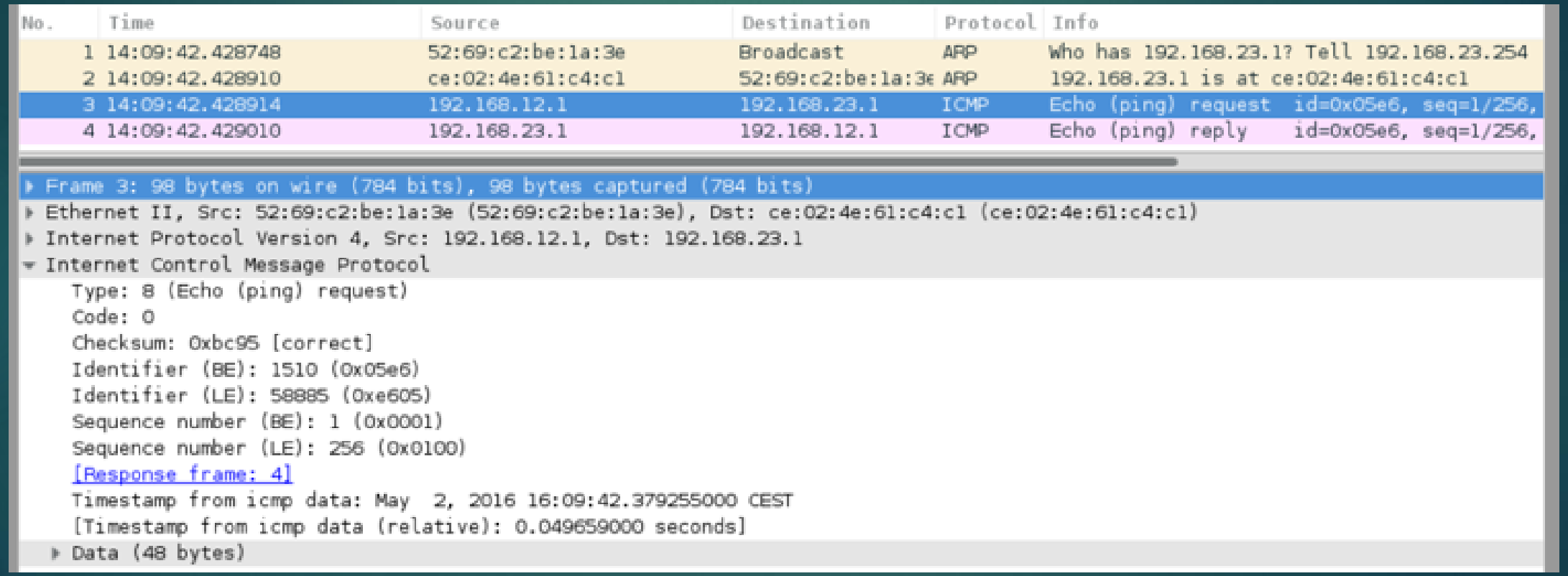
Version (4 bits)	Longueur d'en-tête (4 bits)	Type de service (8 bits)	Longueur totale (16 bits)
Identification (16 bits)		Drapeau (3 bits)	Décalage fragment (13 bits)
Durée de vie (8 bits)	Protocole (8 bits)	Somme de contrôle en-tête (16 bits)	
Adresse IP source (32 bits)			
Adresse IP destination (32 bits)			
Données			



Encapsulation : message ping



ICMP sur Wireshark



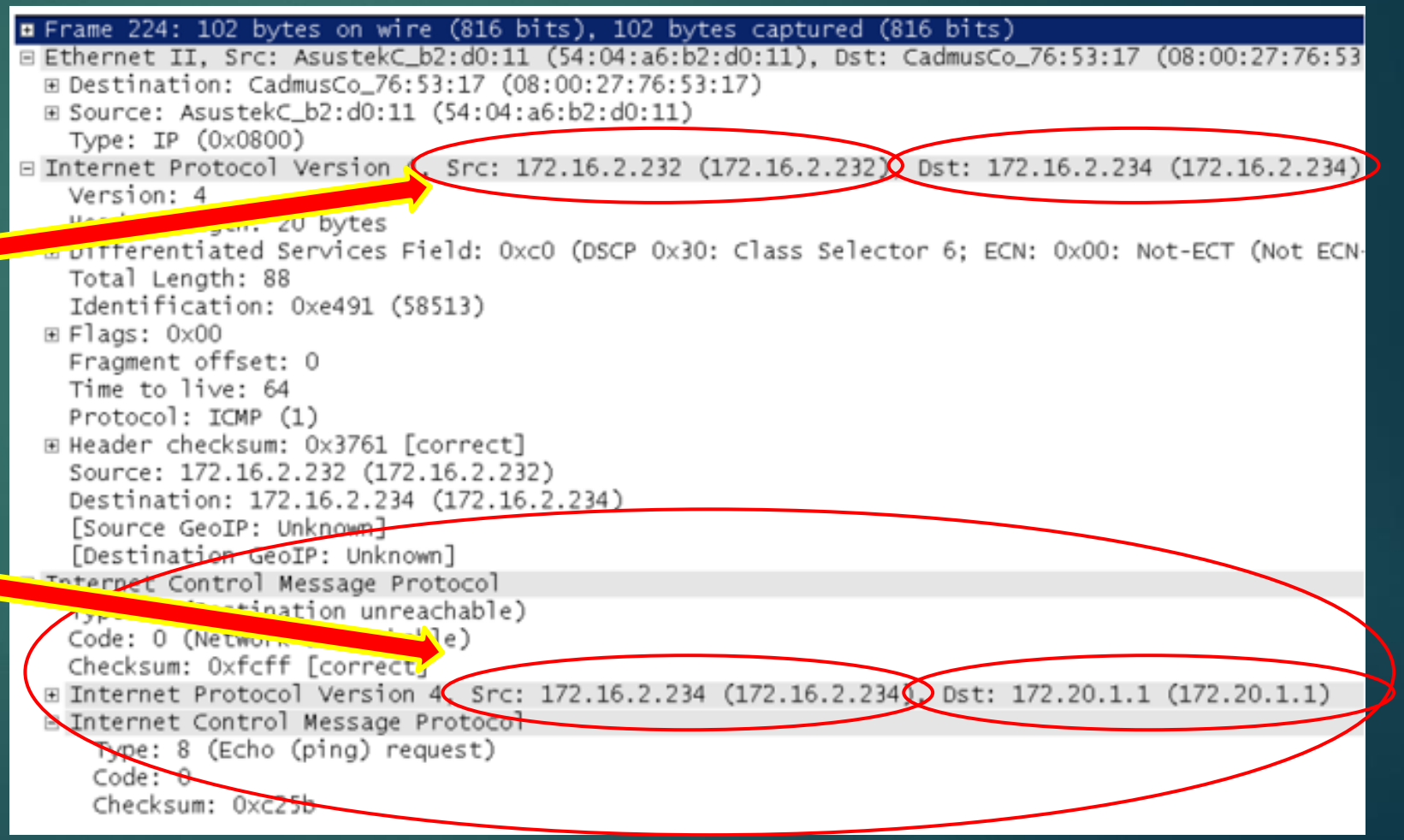
No.	Time	Source	Destination	Protocol	Info
1	14:09:42.428748	52:69:c2:be:1a:3e	Broadcast	ARP	Who has 192.168.23.1? Tell 192.168.23.254
2	14:09:42.428910	ce:02:4e:61:c4:c1	52:69:c2:be:1a:3e	ARP	192.168.23.1 is at ce:02:4e:61:c4:c1
3	14:09:42.428914	192.168.12.1	192.168.23.1	ICMP	Echo (ping) request id=0x05e6, seq=1/256,
4	14:09:42.429010	192.168.23.1	192.168.12.1	ICMP	Echo (ping) reply id=0x05e6, seq=1/256,

```
▸ Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▸ Ethernet II, Src: 52:69:c2:be:1a:3e (52:69:c2:be:1a:3e), Dst: ce:02:4e:61:c4:c1 (ce:02:4e:61:c4:c1)
▸ Internet Protocol Version 4, Src: 192.168.12.1, Dst: 192.168.23.1
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xbc95 [correct]
    Identifier (BE): 1510 (0x05e6)
    Identifier (LE): 58885 (0xe605)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Response frame: 4]
    Timestamp from icmp data: May  2, 2016 16:09:42.379255000 CEST
    [Timestamp from icmp data (relative): 0.049659000 seconds]
  ▸ Data (48 bytes)
```

Un message ICMP

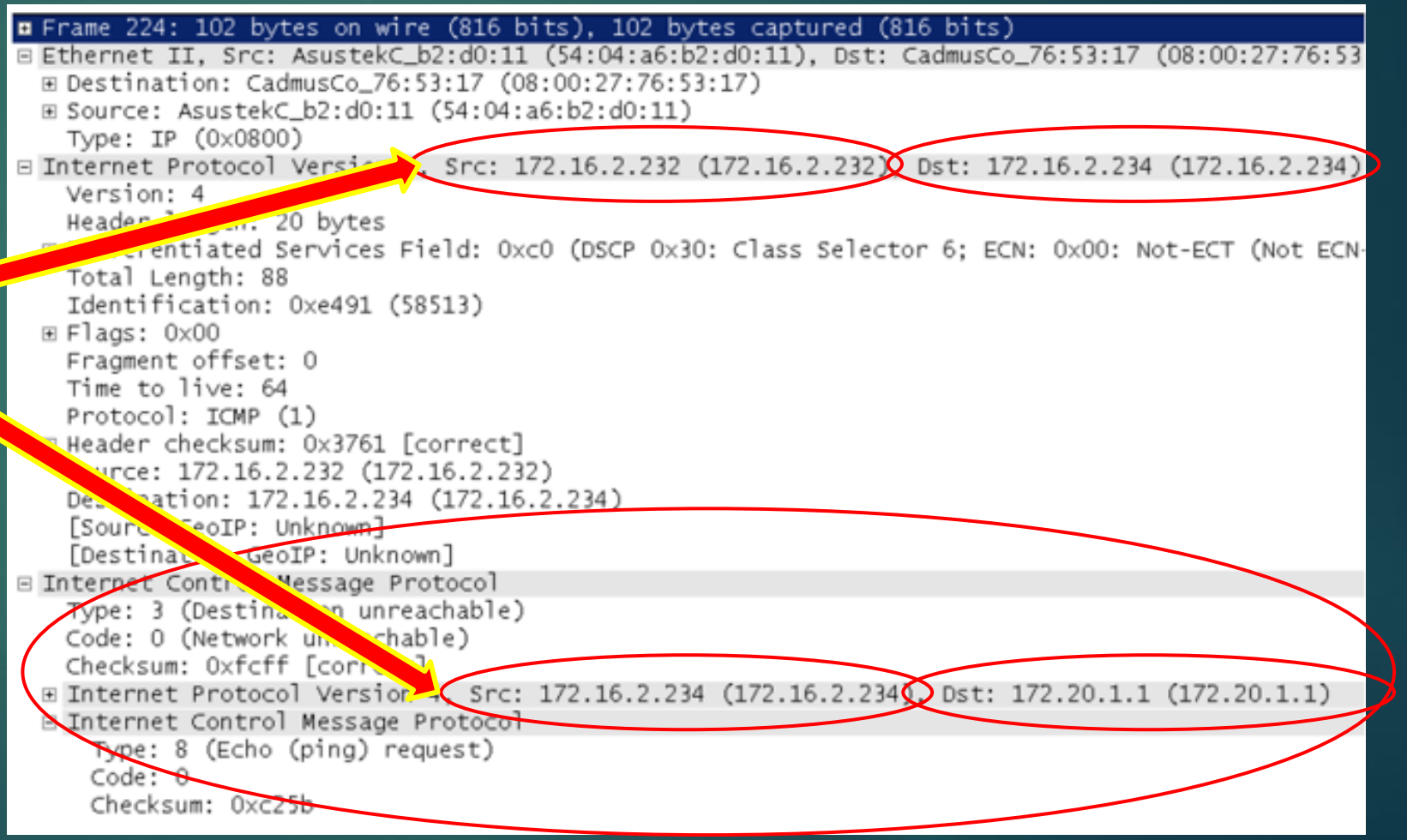
- Source/destination du message ICMP
- Source/destination du message encapsulé par ICMP

Pourquoi une différence ?



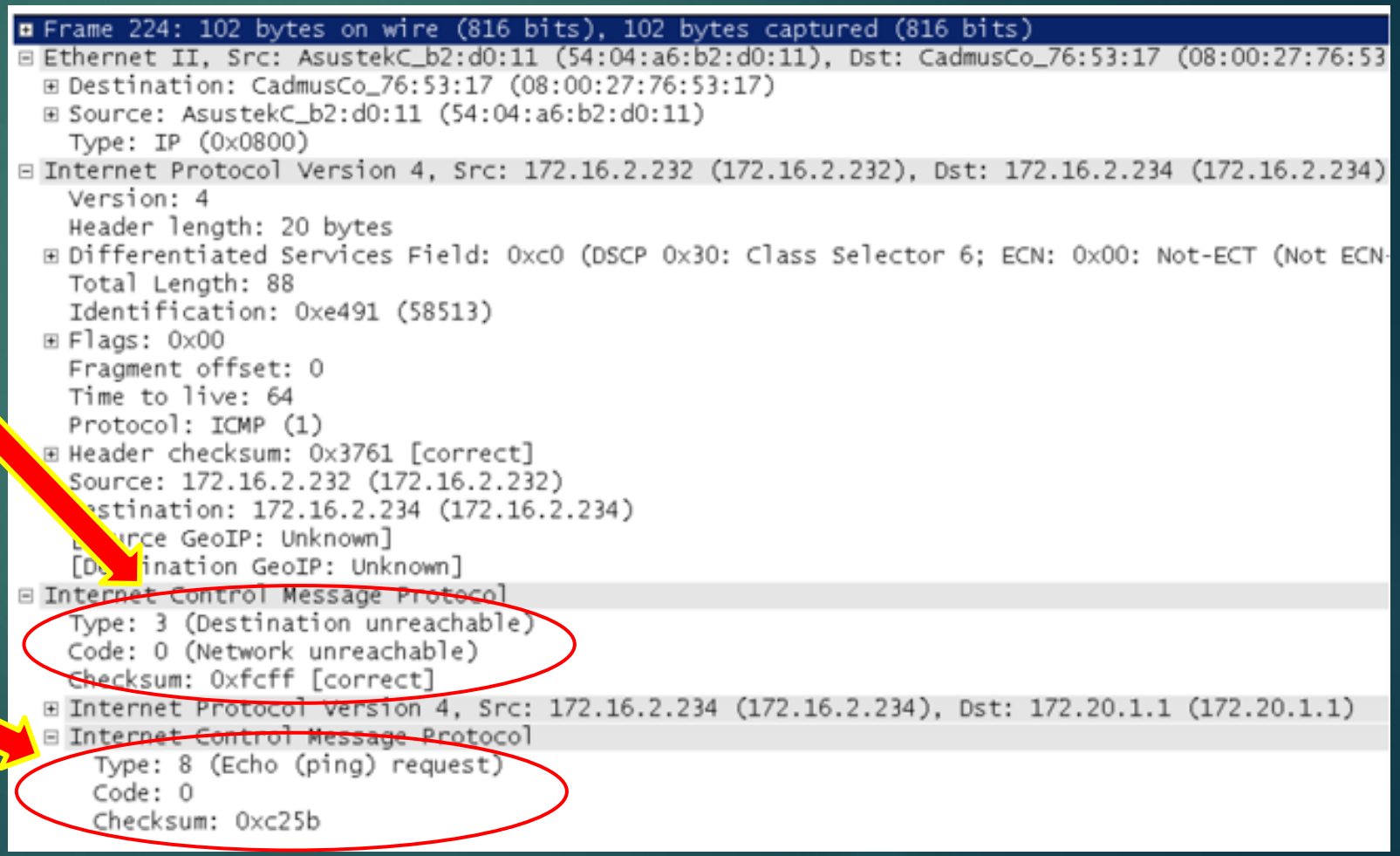
Un message ICMP

- 172.16.2.234 veut envoyer un message (ping) à 172.20.1.1
- 172.20.1.1 est dans un autre réseau
- 172.16.2.232 est alors une passerelle



Un message ICMP

- Type et code de l'erreur signallée par ICMP
- Message qui a causé le problème



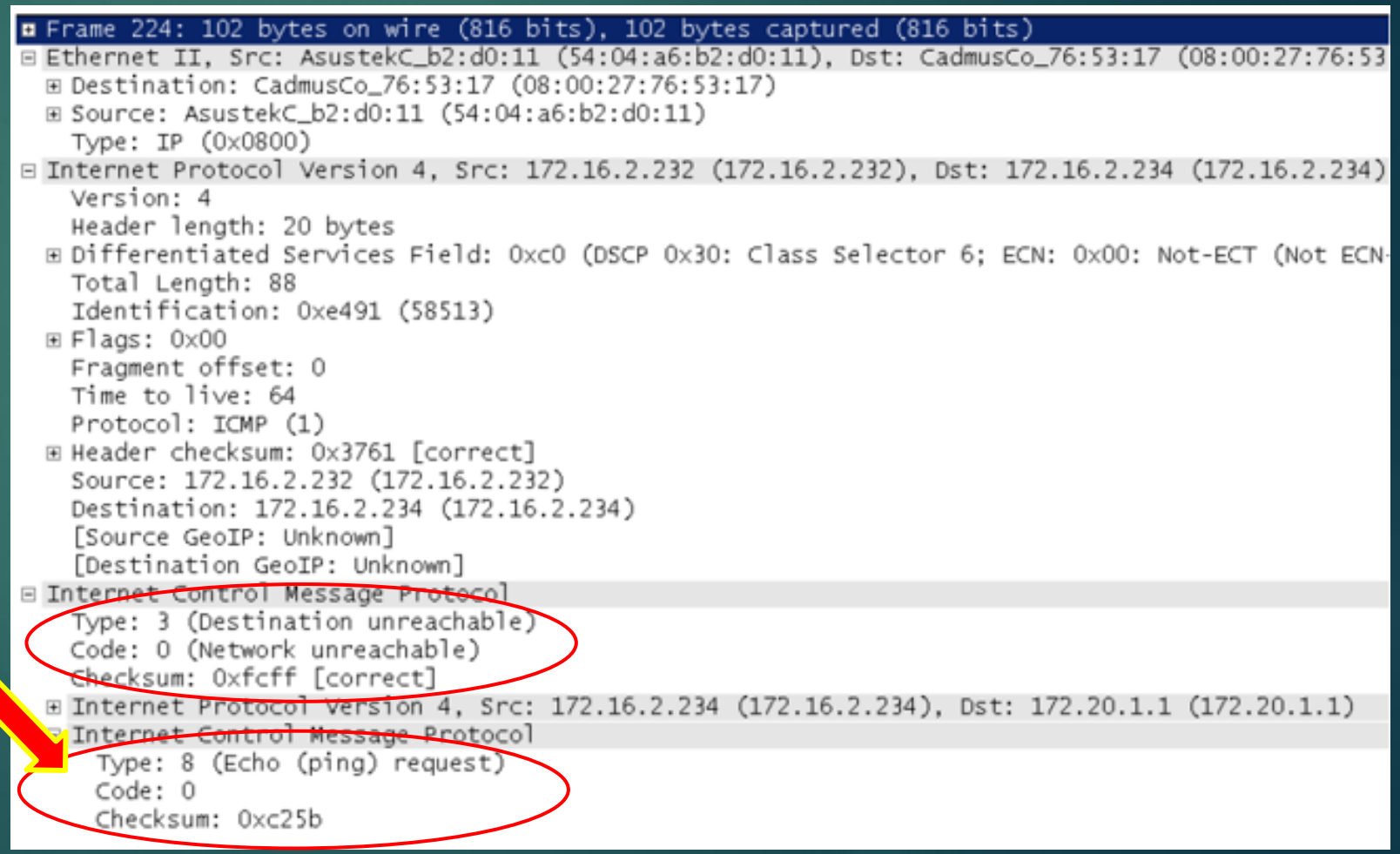
Un message ICMP

- Problème :

  Ping envoyé par

  172.16.2.234

On n'a pas pu contacter ce réseau !



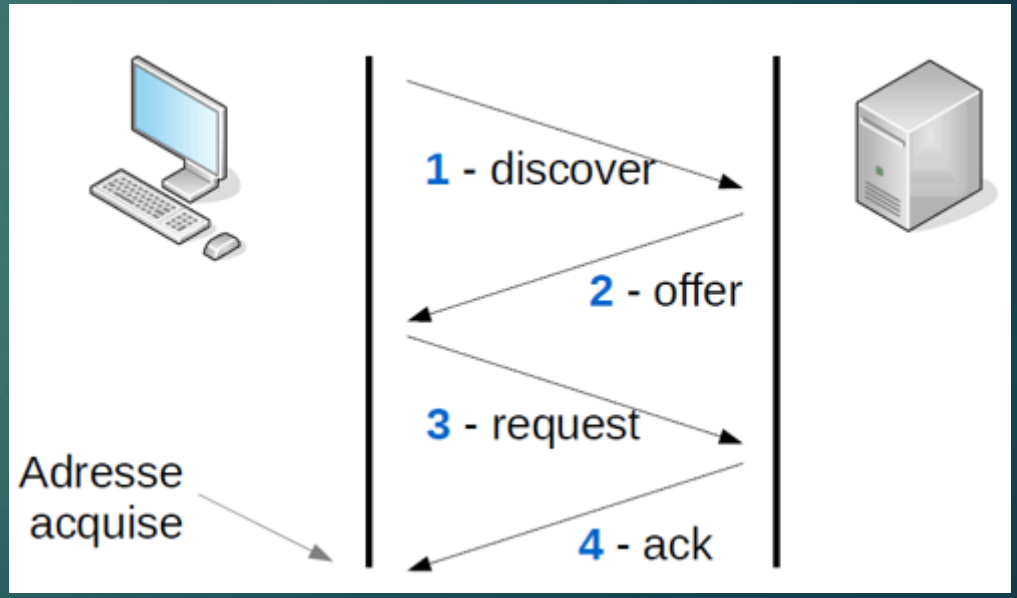
Le protocole DHCP

Problématique

- On arrive dans un nouveau environnement
  - Un autre campus, un hôtel, la maison d'un ami
- On veut se connecter à son réseau
  - Une fois connectés, nous allons faire partie de son réseau
  - L'adresse physique ne change pas, mais on a une nouvelle adresse IP
- Pour la plupart on choisit de le faire automatiquement

DHCP

- Un serveur qui distribue automatiquement la configuration IP :
  - Adresse IP + masque réseau
  - Une passerelle par défaut pour le routage
  - Mise à jour du serveur DNS
- C'est l'ISP qui s'en occupe
  - Ceci marche dans tout réseau locale
  - Gestion centralisée du réseau
- 4 étapes



Intermezzo : protocole client-serveur

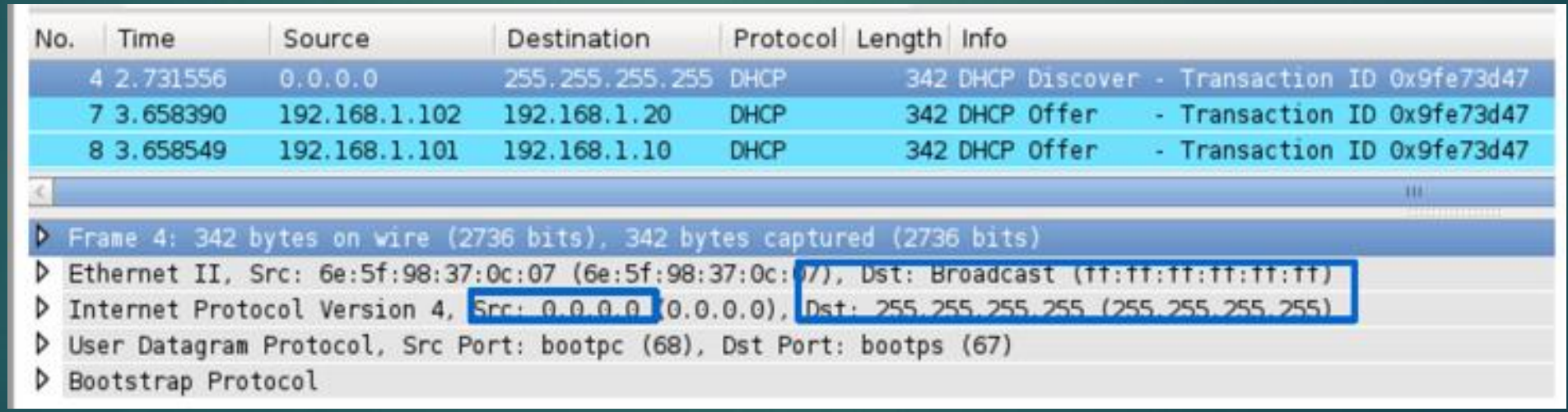
Client, serveur

- Protocole client-serveur :
  - Les messages passent entre une machine client et une machine serveur
- Machine serveur :
  - Toujours à l'écoute, souvent sur un port standard (ss -ln)
  - Peut se connecter à un client (ss -tun ou ss -tuan)
  - Peut jouer également le rôle d'un client
- Machine client :
  - Contacte le serveur (typiquement port non-standard)

Fin Intermezzo

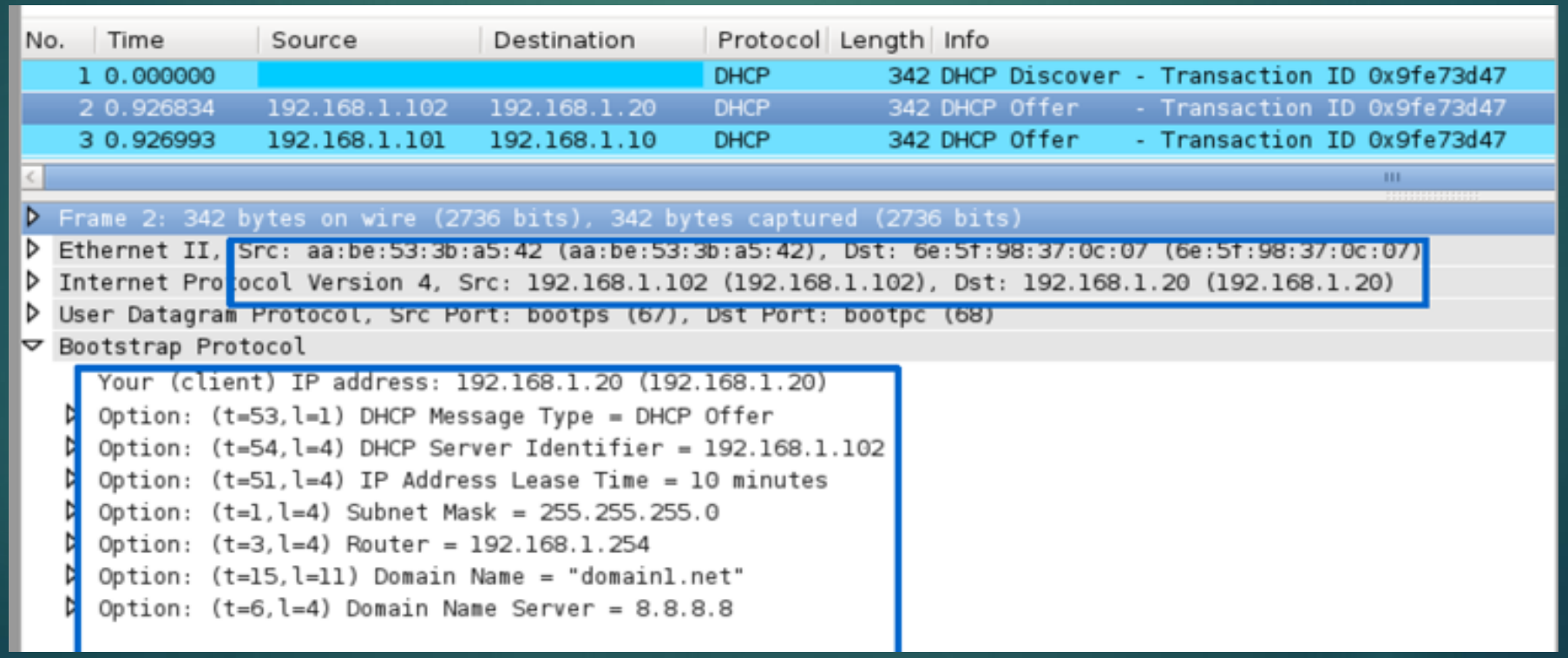
Etape 1 : Discover

- Le nouveau ordinateur veut recevoir une adresse IP
  - Il ne connaît pas son adresse IP
  - Il fait un broadcast



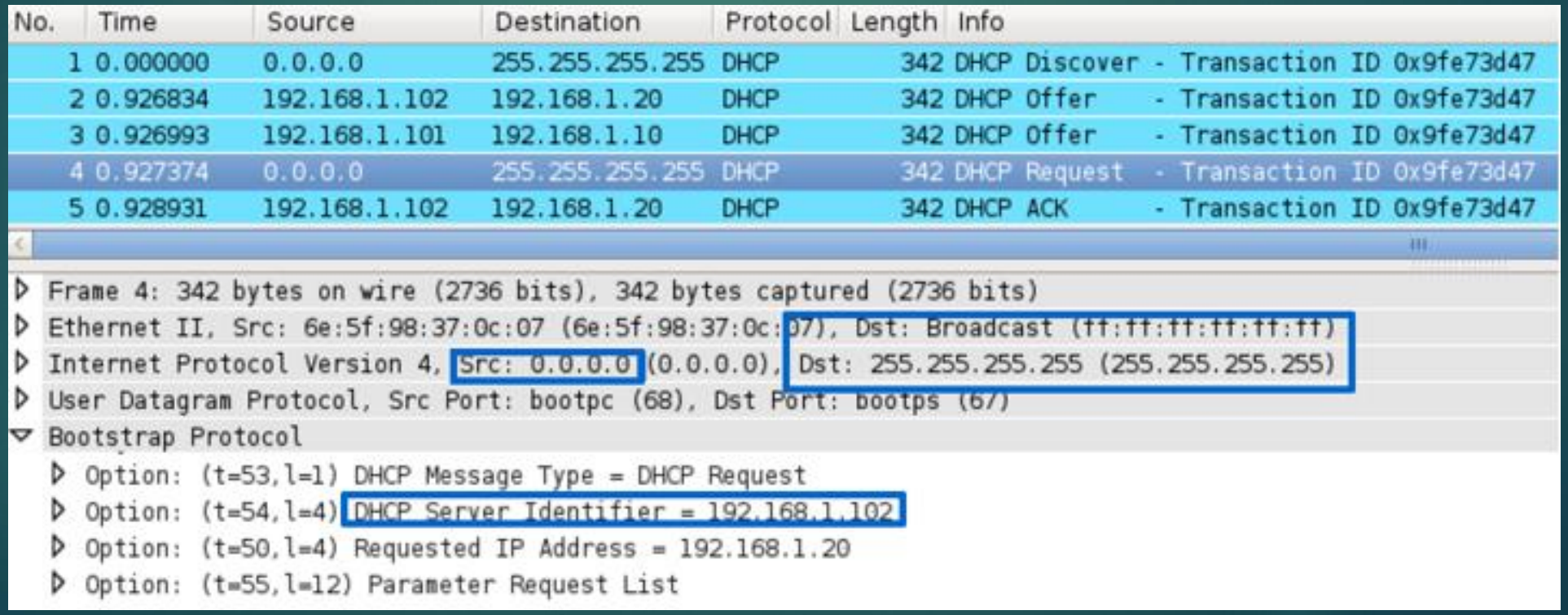
Etape 2 : Offer

- Le serveur DHCP a reçu son message et lui propose une adresse IP
  - Cette offerte est limitée pour un nombre de minutes initiellement



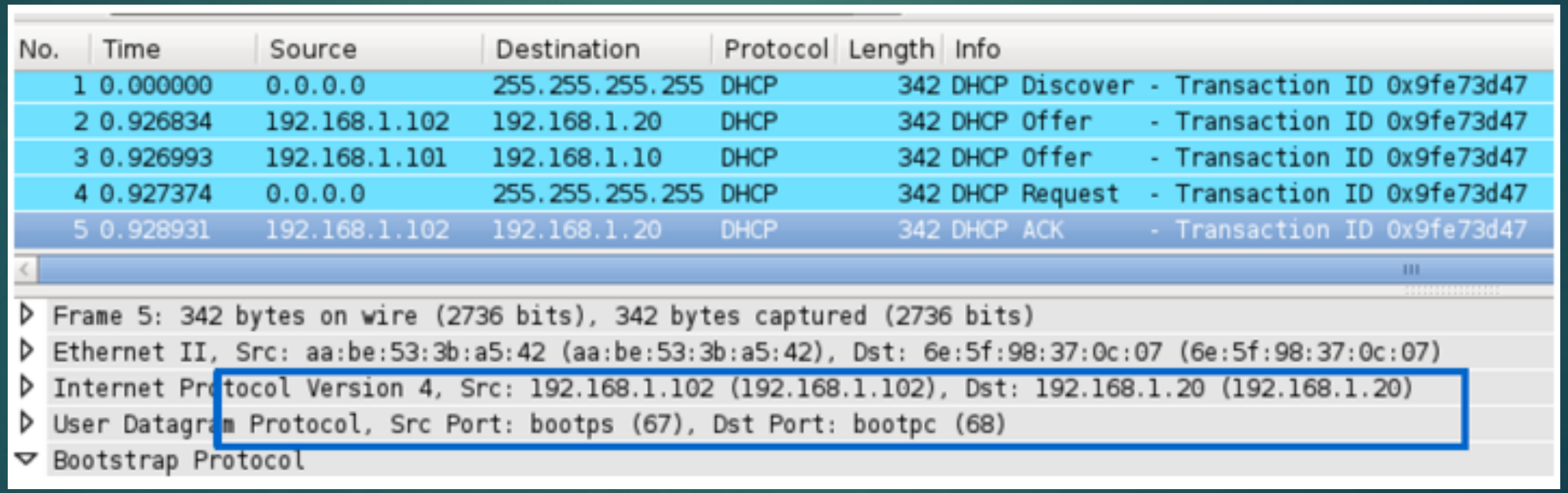
Etape 3 : Request

- Le nouvel ordinateur choisit de demander l'adresse qu'on lui a donné
  - L'adresse ne va plus expirer



Etape 4 : Acknowledgement

- Le serveur confirme l'allocation de la nouvelle config. IP



Configuration DHCP

- Côté serveur :
  - Configuration fichier dhcpd.conf
    - Spécifier sur quelle interface on fiat DHCP
    - Spécifier : plage d'adresses à utiliser, optionnellement quel routeur à utiliser, DNS...
- Côté client :
  - Obtenir une adresse dynamiquement : dhclient eth0
  - Configuration perenne : fichier /etc/network/interfaces

Fichier interfaces

- Configuration perenne des interfaces d'une machine

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.56.11
netmask 255.255.255.0
gateway 192.168.56.1

auto eth1
iface eth1 inet dhcp
```

Loopback, interface lo

Config. statique, eth0

Config. par DHCP

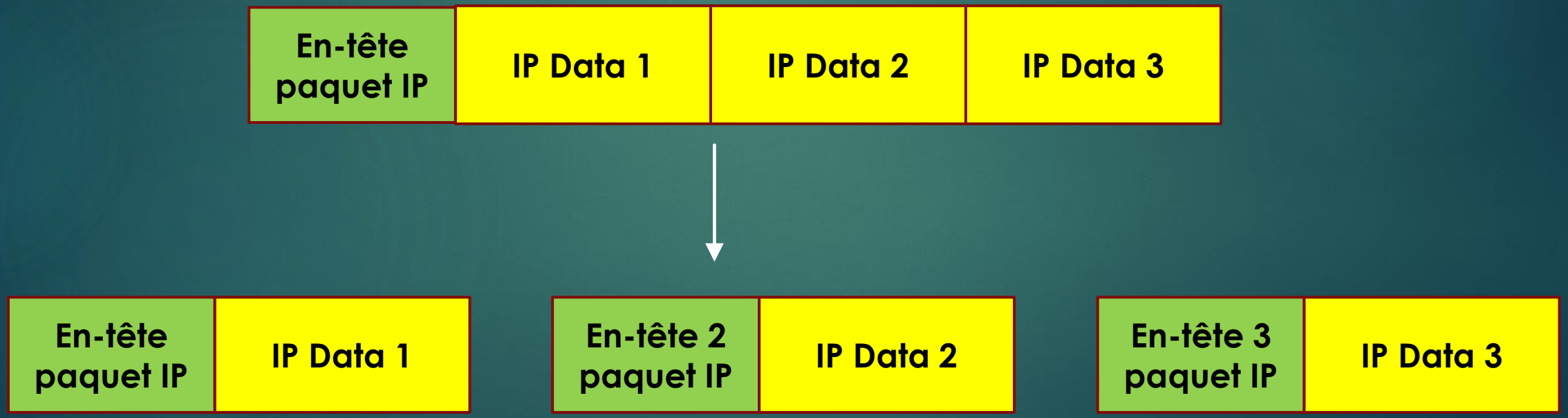
Différences DHCP, interfaces

- Le fichier /etc/network/interfaces indique :
  - L'adresse IP + routeur de la machine elle-même
  - Pas de possibilité de configuration DNS
- Le fichier /etc/dhcp/dhcpd.conf indique :
  - Une plage d'adresses IP qui seront données à une machine cherchant une configuration dynamique
  - Un routeur que les machines prenant une config. dynamique peuvent utiliser
  - Un serveur DNS pour ces machines aussi ...

Le protocole IP

Les paquets IP

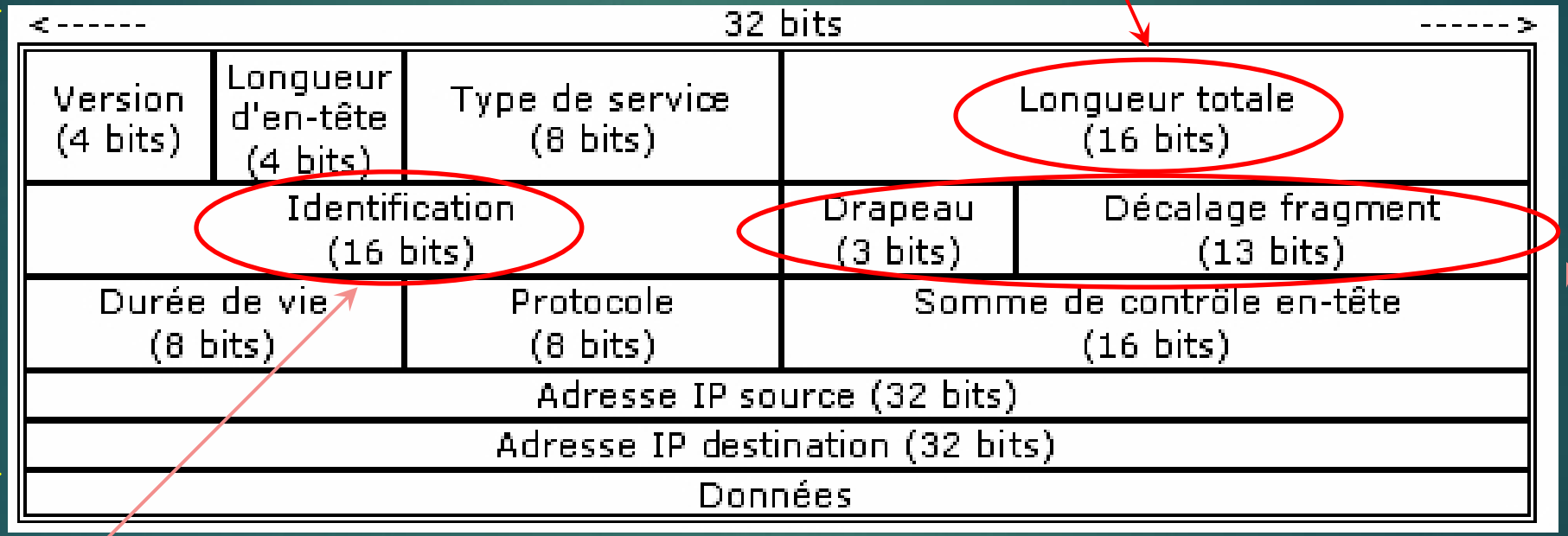
- La taille maximale d'un paquet IP est 1480 octets
- Qu'est-ce qui se passe si on veut envoyer un message plus gros ?



Les en-têtes indiquent la suite de messages

Structure d'un paquet IP

Taille totale: 20 + longueur données



20 octets

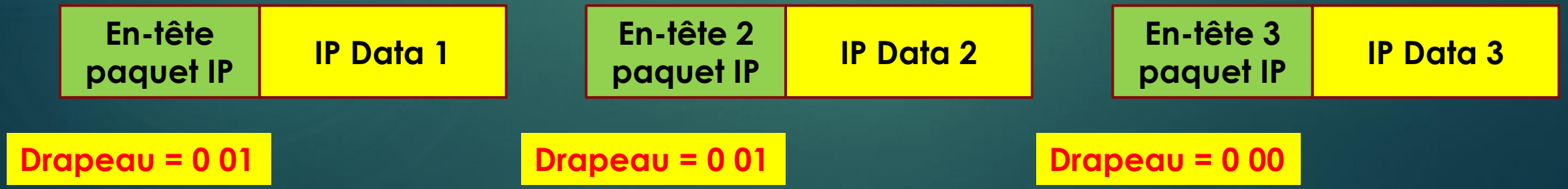
Source: linux-France.org

Même valeur pour les fragments du même message : identifie le message

Valeurs différentes pour différents fragments

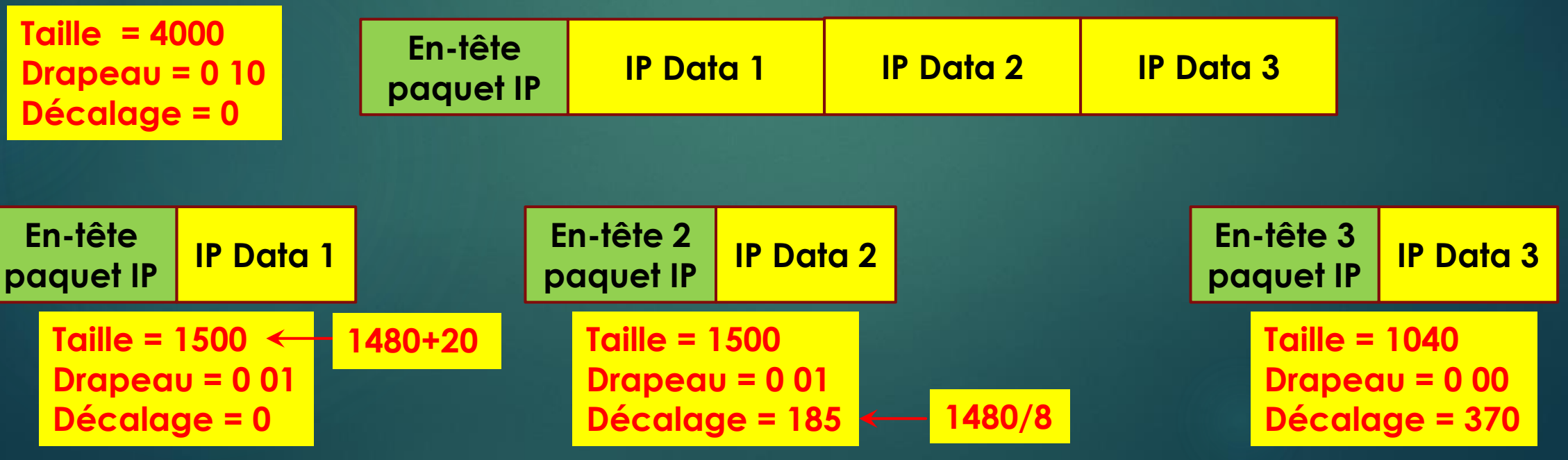
Le drapeau

- Le drapeau indique le début/milieu/la fin du message : 3 bits :
  - Premier bit est réservé
  - Deuxième bit : Don't Fragment
    - Si ce bit = 1, alors on n'accepte que le premier fragment d'un paquet
    - A utiliser si on ne veut envoyer qu'un paquet
  - Troisième bit : More Fragments
    - Si ce bit = 1, alors on attend plus de fragments. Si ce bit = 0, dernier fragment

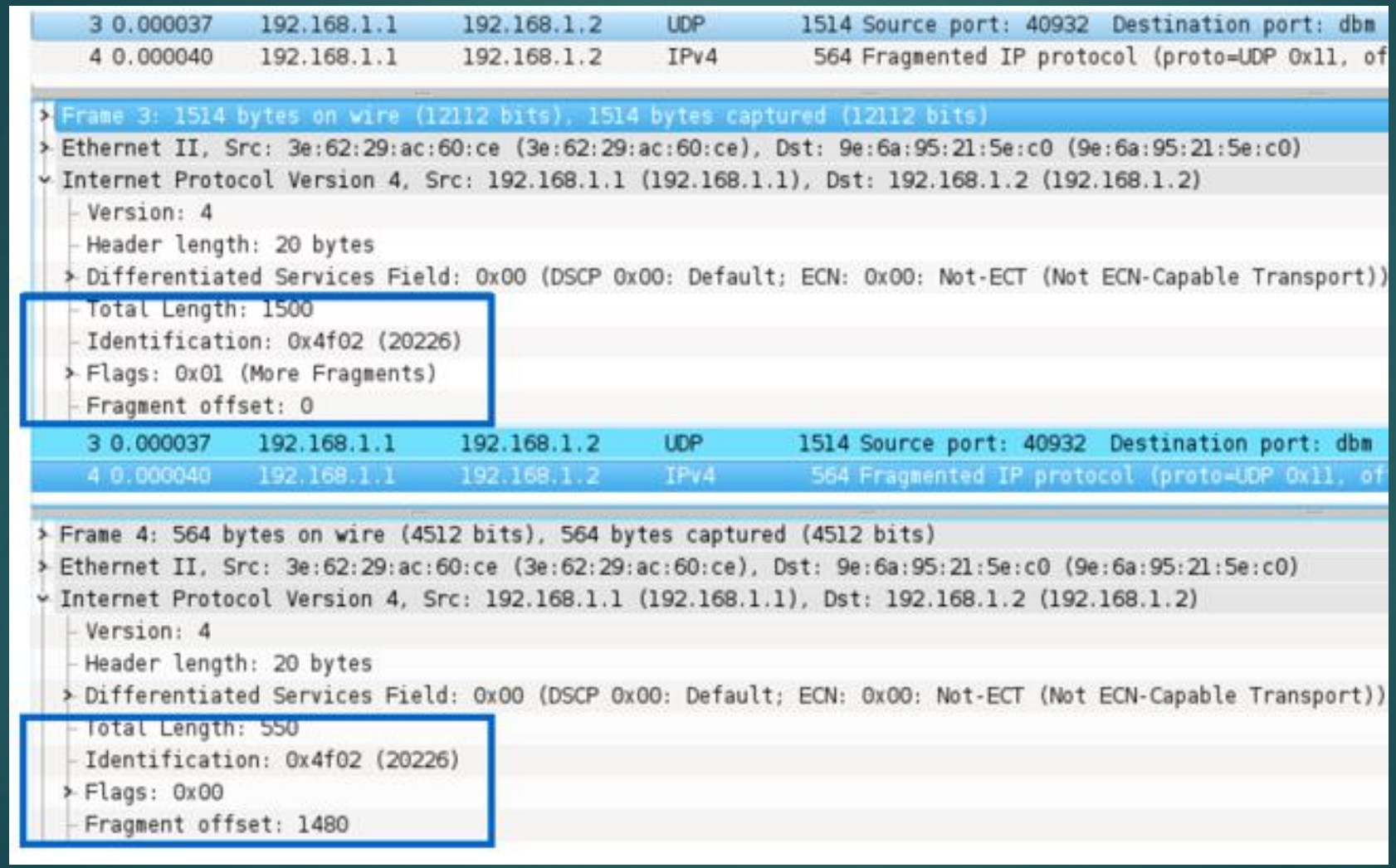


Le décalage

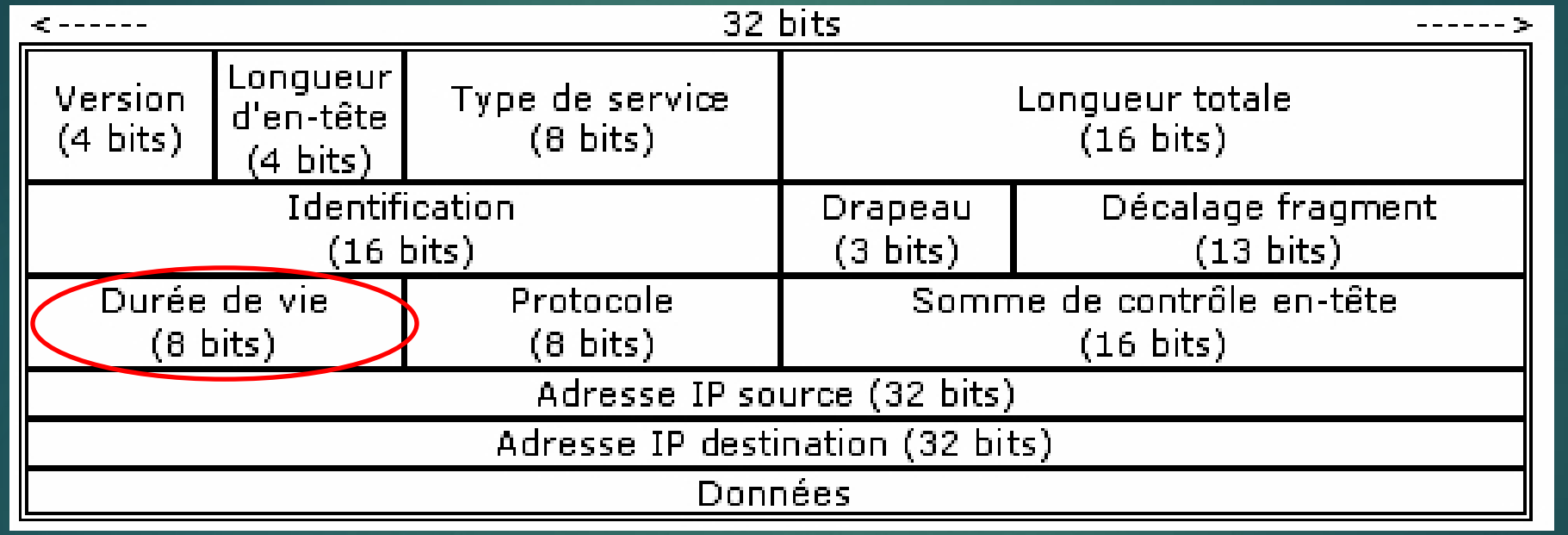
- Le décalage indique l'ordre des paquets dans un message
  - Les premiers paquets seront saturés (1480 octets + 20 octets en-tête)
  - Le dernier peut avoir une taille plus petite
  - Taille décalage sur 13 bits tandis que taille totale sur 16 bits
    - on calcule le decalage divisé par 8



En pratique…


3 0.000037 192.168.1.1 192.168.1.2 UDP 1514 Source port: 40932 Destination port: dbm
4 0.000040 192.168.1.1 192.168.1.2 IPv4 564 Fragmented IP protocol (proto=UDP 0x11, of
Frame 3: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
Ethernet II, Src: 3e:62:29:ac:60:ce (3e:62:29:ac:60:ce), Dst: 9e:6a:95:21:5e:c0 (9e:6a:95:21:5e:c0)
Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 1500
Identification: 0x4f02 (20226)
Flags: 0x01 (More Fragments)
Fragment offset: 0
3 0.000037 192.168.1.1 192.168.1.2 UDP 1514 Source port: 40932 Destination port: dbm
4 0.000040 192.168.1.1 192.168.1.2 IPv4 564 Fragmented IP protocol (proto=UDP 0x11, of
Frame 4: 564 bytes on wire (4512 bits), 564 bytes captured (4512 bits)
Ethernet II, Src: 3e:62:29:ac:60:ce (3e:62:29:ac:60:ce), Dst: 9e:6a:95:21:5e:c0 (9e:6a:95:21:5e:c0)
Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 550
Identification: 0x4f02 (20226)
Flags: 0x00
Fragment offset: 1480

Un autre champ intéressant



Pourquoi une durée de vie ?

- Dans un réseau, les routeurs envoient des messages entre eux
  - ... ou vers un sous-réseau
- L'idée est de trouver le destinataire
- Mais parfois il y a des problèmes :
  - Les tables de routage ne permettent pas accès à ce destinataire
  - Le destinataire n'existe pas (ou n'existe plus)
  - ...

Comment s'assurer que les paquets ne vont pas rester dans le réseau pour toujours ?

TTL = Time to live

- Durée de vie d'un paquet = # de sauts (hops) dans le réseau
- Chaque fois qu'un router fait suivre un paquet = 1 saut
- Si la durée de vie arrive à 0, message ICMP à la source

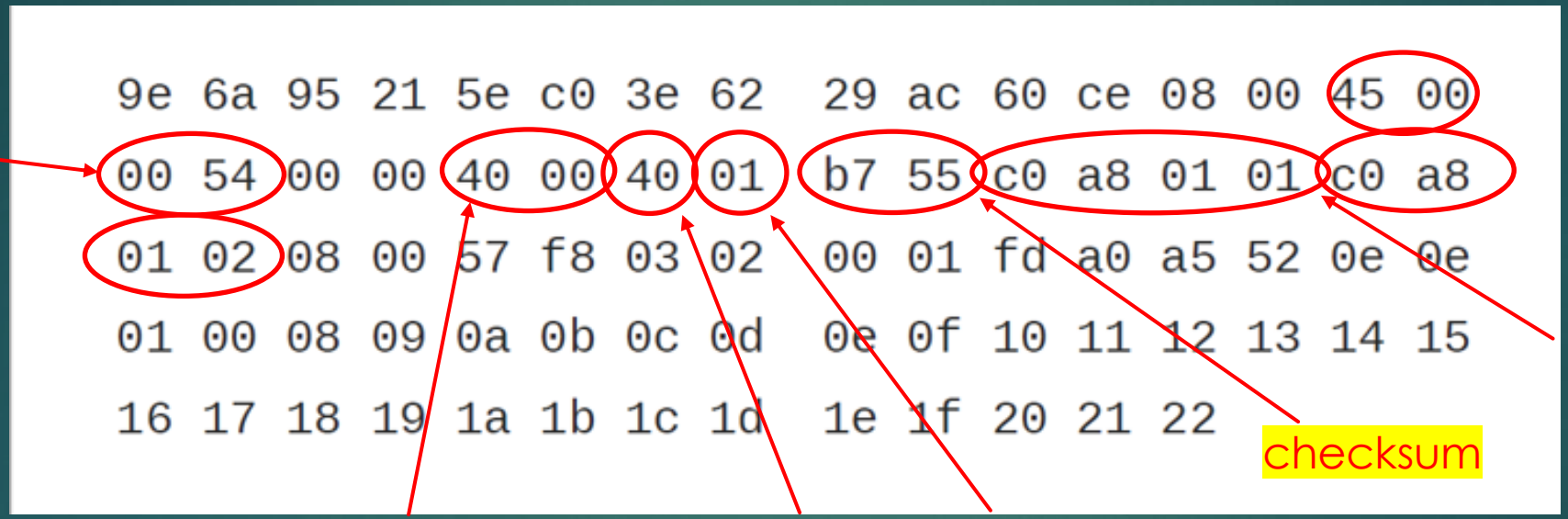
Lire un en-tête IP

En-tête MAC : Dest

En-tête MAC : Src

Ethernet

IPv4
5 octets header
Pas de service spéciaux



taille totale

½ IP dest

½ IP dest

IP src

0100 0000 0x00

drapeau don't fragment

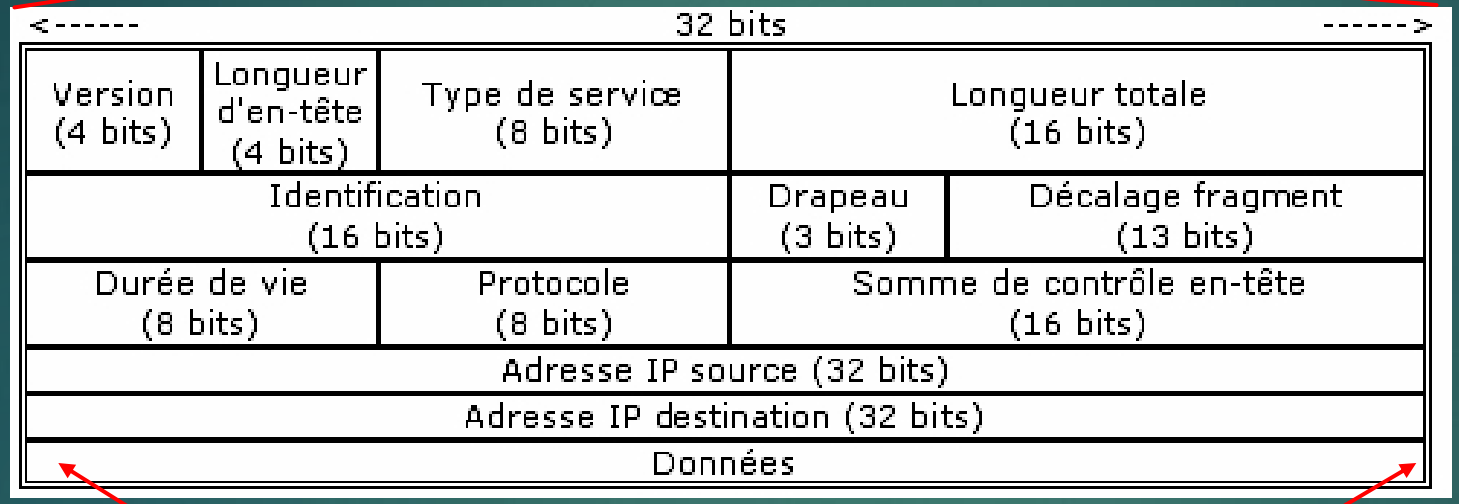
offset 0

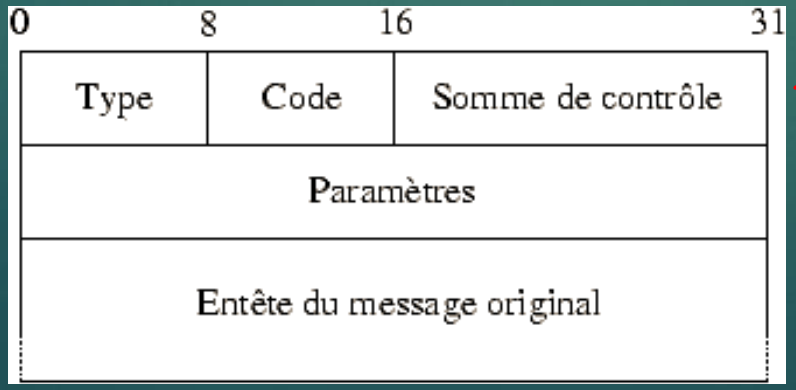
TTL = 64

Protocole: ICMP

Rappel : Structure paquet IP

adressesMAC typeProt Paquet IP





Lire un en-tête IP


```
9e 6a 95 21 5e c0 3e 62  29 ac 60 ce 08 00 45 00
00 54 00 00 40 00 40 01  b7 55 c0 a8 01 01 c0 a8
01 02 08 00 57 f8 03 02  00 01 fd a0 a5 52 0e 0e
01 00 08 09 0a 0b 0c 0d  0e 0f 10 11 12 13 14 15
16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22
```

checksum

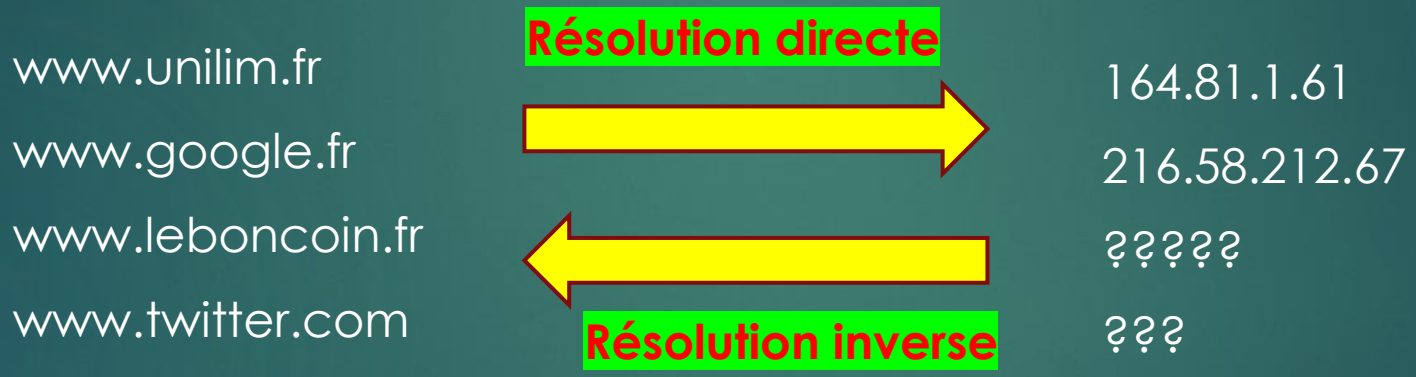
type :
08 = ping request

réseau non-joignable

Le protocole DNS

Problématique

- Pour accéder à un site Internet, on rentre une adresse URL
  - Comment notre ordinateur sait-il à quelle adresse IP se connecter ?



- Solution à l'ancienne : **fichiers hosts**
- Solution intermédiaire : **system NetBIOS**
- Solution moderne : **DNS**

A l'ancienne : fichier hosts

- Un fichier texte, à copier sur chaque machine
  - Contient la correspondence entre les adresses IP et les noms (e.g. URL)
  - Marchait sur Windows & Linux
  - La modification était manuelle

```
127.0.0.1       localhost
192.168.0.20    IUTINFO-VPN-04.iut.unilim.fr
192.168.0.5     hercule hercule.iut.unilim.fr
```

Solution moderne : DNS

- DNS = Domain naming system
- Structure en arbre : permet une hiérarchie de noms
  - Par exemple www.google.com et mail.google.com
- DNS = un annuaire distribué
  - Si un serveur DNS n'a pas la réponse, il renvoie sa demande à un autre serveur, et cela continue jusqu'à avoir une réponse
  - La mise à jour est automatique
- Protocole DNS : client-serveur, marche sur TCP ou UDP

Les fichiers DNS et hosts aujourd'hui

- Le fichier /hosts :
  - aujourd'hui très simpliste, sert à identifier le localhost par exemple

/etc/hosts

```
root@debian7-basic:~/Perso# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       debian7-basic

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

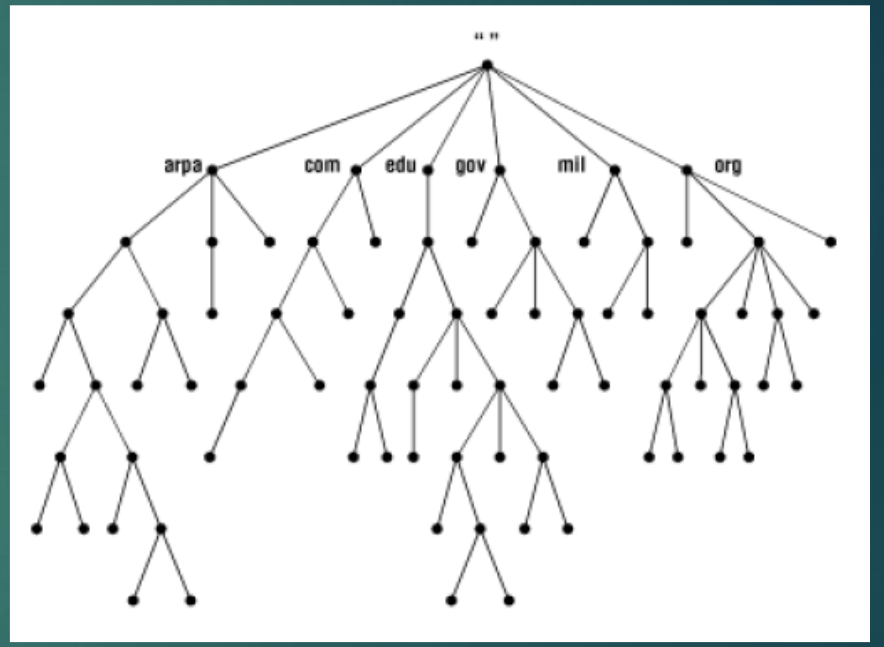
- Le fichier resolv.conf : plusieurs fonctionnalités
  - indique par exemple le domaine, où chercher des adresses
  - Indique quel(s) serveur(s) DNS à utiliser

/etc/resolv.conf

```
root@debian7-basic:~/Perso# cat /etc/resolv.conf
# Generated by NetworkManager
domain unilim.fr
search unilim.fr
nameserver 164.81.1.4
nameserver 164.81.1.5
```

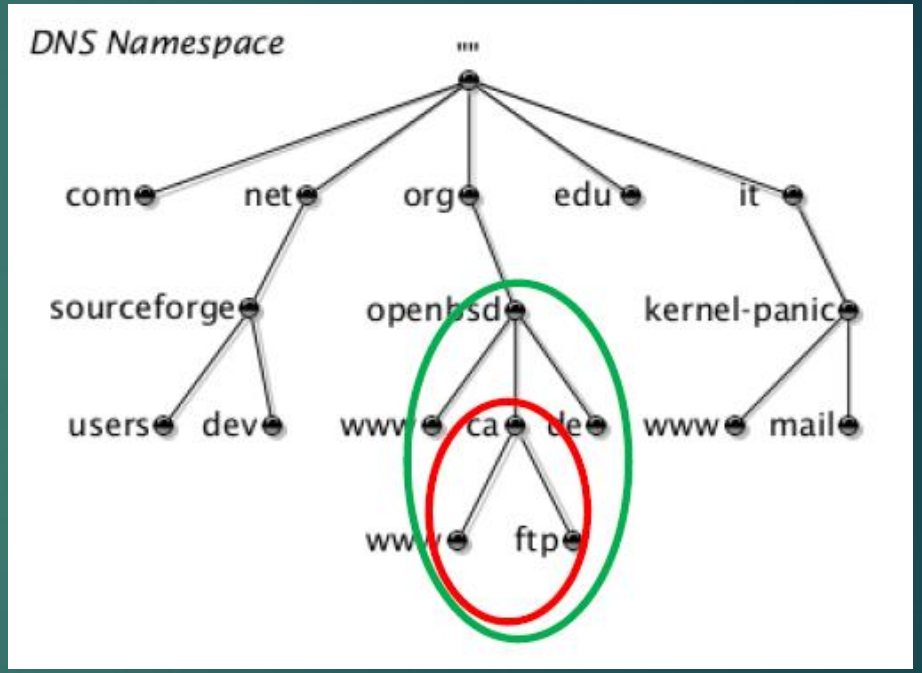
Les noms de domaine

- Namespace : l'espace de nommage
- Arbre de noms avec racine commune
  - jusqu'à 127 niveaux (max 253 char ASCII)
  - Chaque noeud à un nom...
  - ... sauf la racine: "."
- Le nom d'un domaine se reconstitue en ordre inverse, du nom du noeud au plus en bas vers celui le plus en haut (finissant par .)



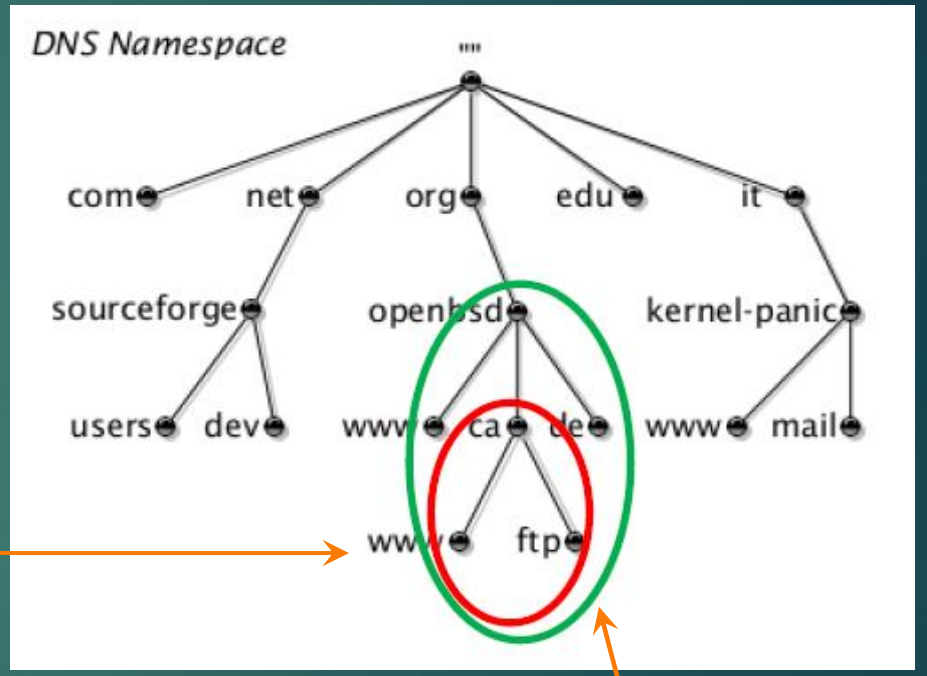
Exemple

- Domaine en rouge : ca.openbsd.org.
  - Avec deux machines :
    - www.ca.openbsd.org
    - ftp.ca.openbsd.org
- Domaine en vert : openbsd.org.
  - plusieurs sous-domaines



Exemple

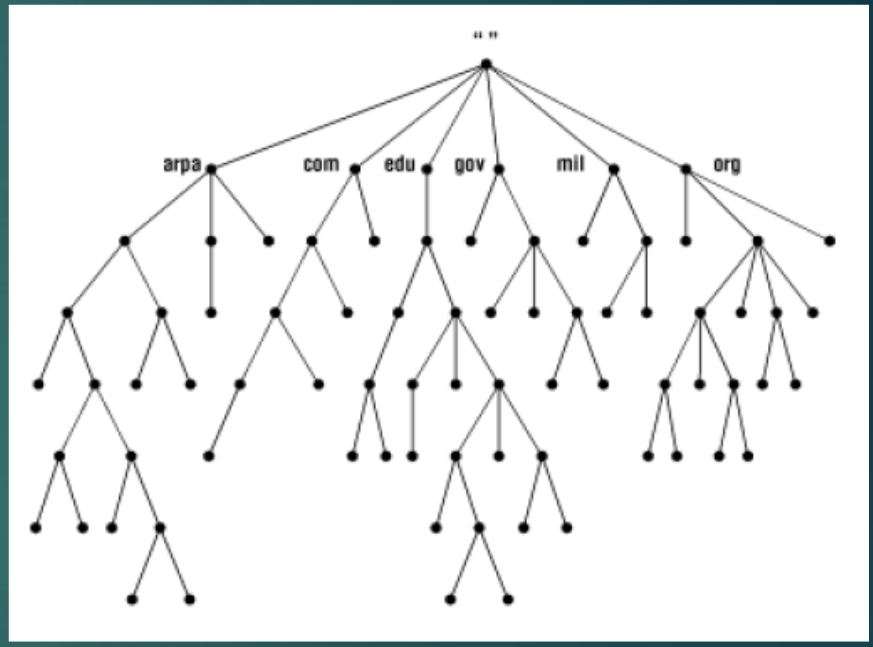
- Domaine en rouge : ca.openbsd.org.
  - Avec deux machines :
    - www.ca.openbsd.org
    - ftp.ca.openbsd.org
- Les pings sur une machine dans un domaine s'interprètent premièrement "localement"



ping www sur machine ftp.ca.openbsd.org

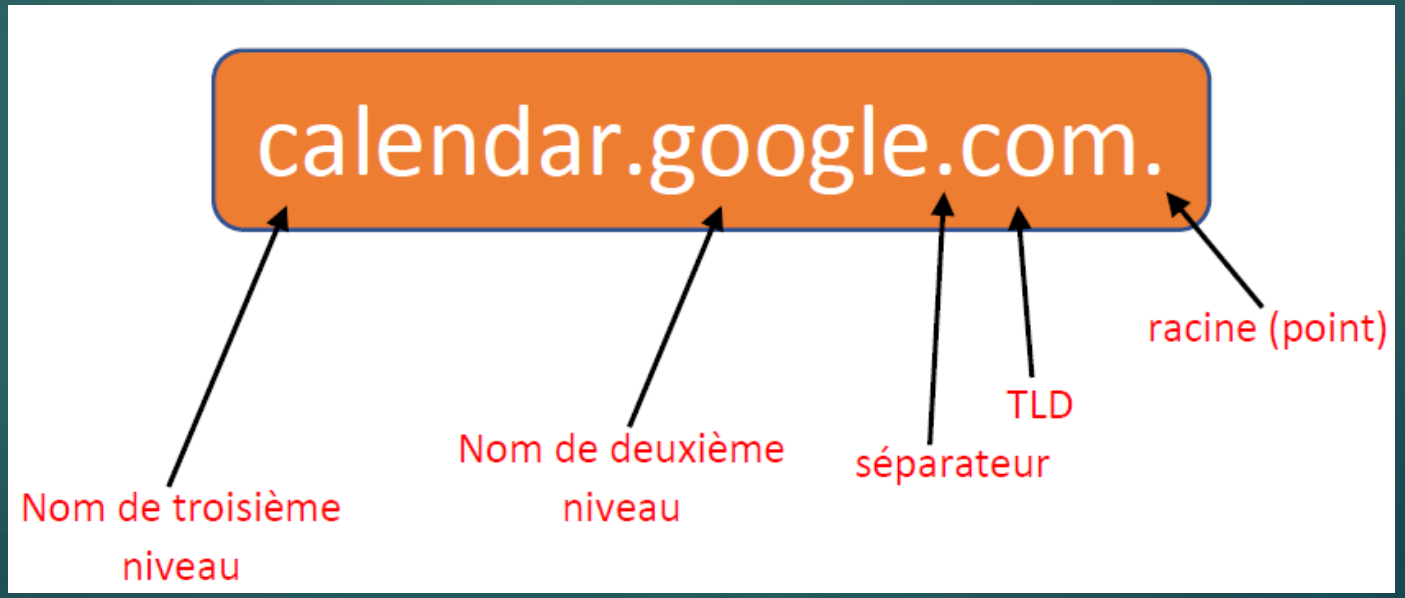
Organisation des noms

- TLD : Top-Level Domain
  - Niveau le plus haut du nom
- Plusieurs catégories :
  - .arpa : réservé, administratif
  - .ca, .be, .fr, etc. : country-code TLD (ccTLD)
  - génériques de 1er niveau : generic TLD : .com, .mil
  - génériques restreint : .biz, .name, ,pro
  - sponsorisés (sTLD) : .post



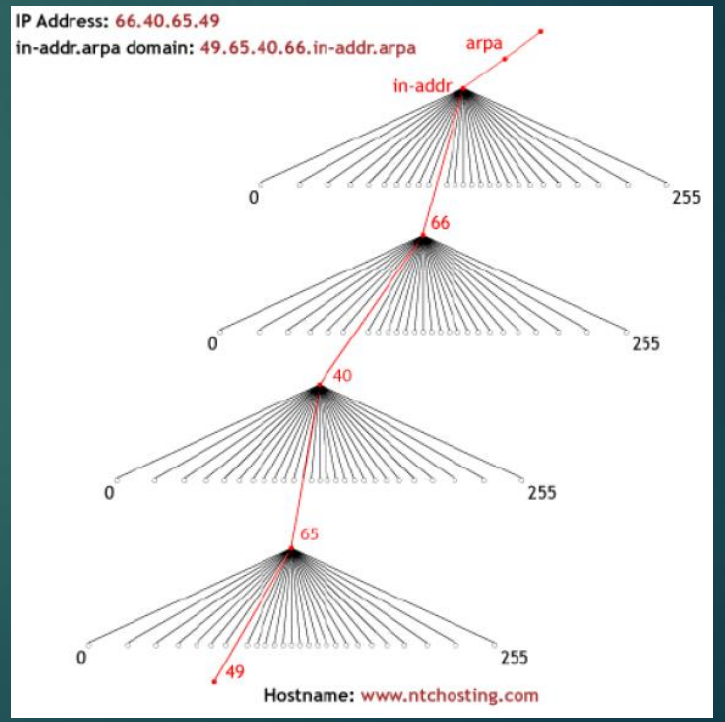
Nom rélatif, nom absolu

- Fully qualified domain name (FQDN) :
  - Nom complet d'une machine, finissant toujours par un point : .
  - La partie la plus significative est à droite, au contraire des adresses IP



La résolution DNS

- Résolution directe : parcourrir l'arbre par ordre inversée de la hiérarchie de son FQDN
- Résolution inverse : domaine spéciale : in-addr.arpa
  - Contient un arbre inversé : le noeud in-addr a 256 noeuds fils
  - Chaque noeud fils a encore 256 noeuds fils...
  - ... etc. (sur 4 niveaux)



Client, serveur DNS

- Client DNS :
  - s'appelle resolver
  - configuration du fichier /etc/resolv.conf : nameserver <@IP serveur DNS>
- Configuration de base d'un serveur DNS :
  - Nous allons configurer seulement une résolution directe de nom
  - Première configuration : spécifier les domaines pour lesquels le serveur fait autorité
  - Ensuite : pour chaque domaine, détailler un fichier de configuration