INF157 - Utilisation des Réseaux

Licence 3 Informatique

Arnaud Pecher (repris par Damien Magoni)

Bureau 322, Bâtiment A30, LaBRI
Université de Bordeaux

Licence 3 Informatique - Bordeaux

Cours 2

Programmation des réseaux (sockets)

1. Sockets
2. Aperçu des sockets en C
3. Sockets en Java
   - UDP
   - TCP

Plan

1. Sockets
2. Aperçu des sockets en C
3. Sockets en Java
   - UDP
   - TCP

Sockets de Berkeley

sockets

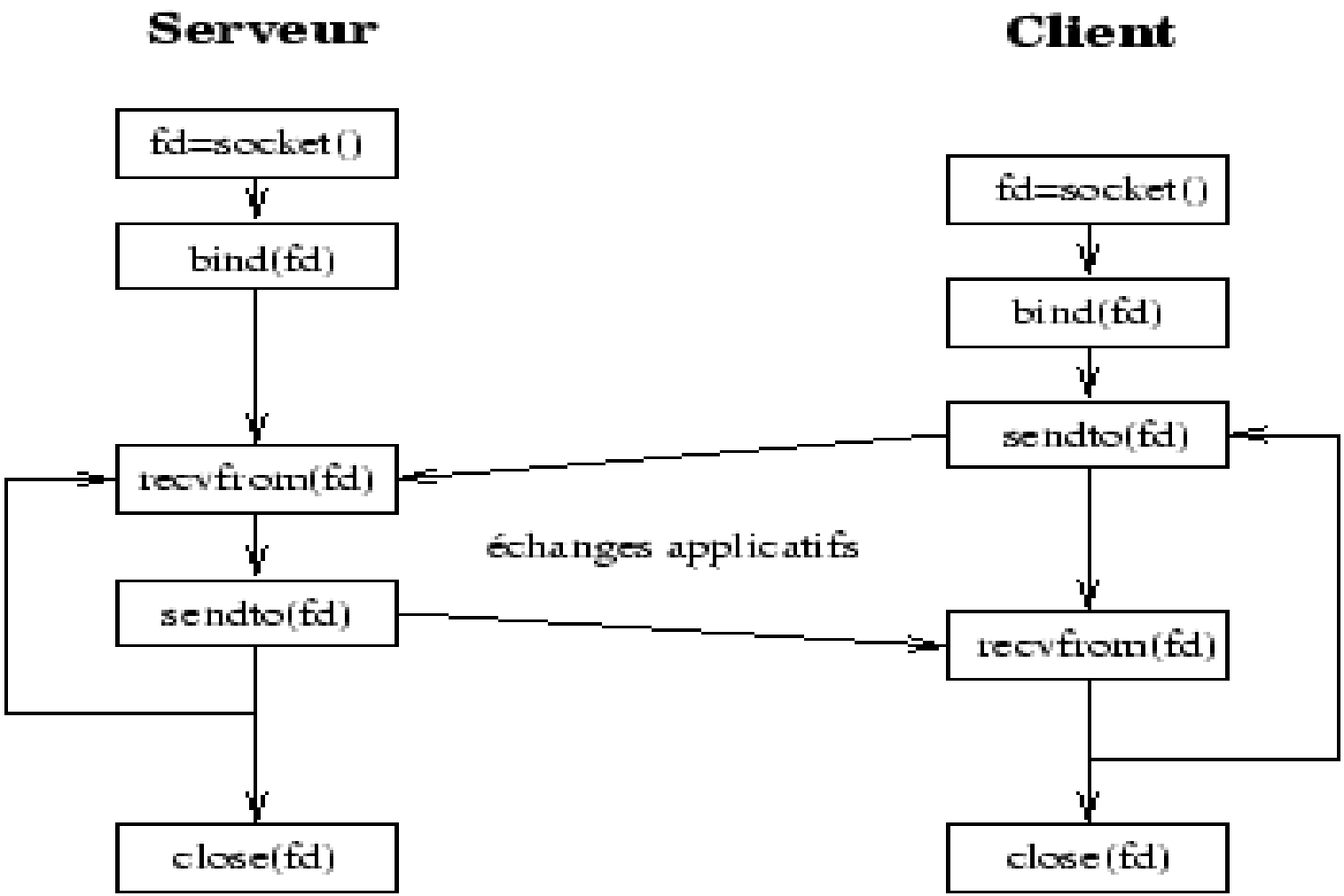
Deux concepts :

- ensemble de primitives : *sockets de Berkeley* ;
- extrémité d'un canal de communication par lequel un processus peut émettre ou recevoir des données.

Plan

1. Sockets
2. Aperçu des sockets en C
3. Sockets en Java
   - UDP
   - TCP

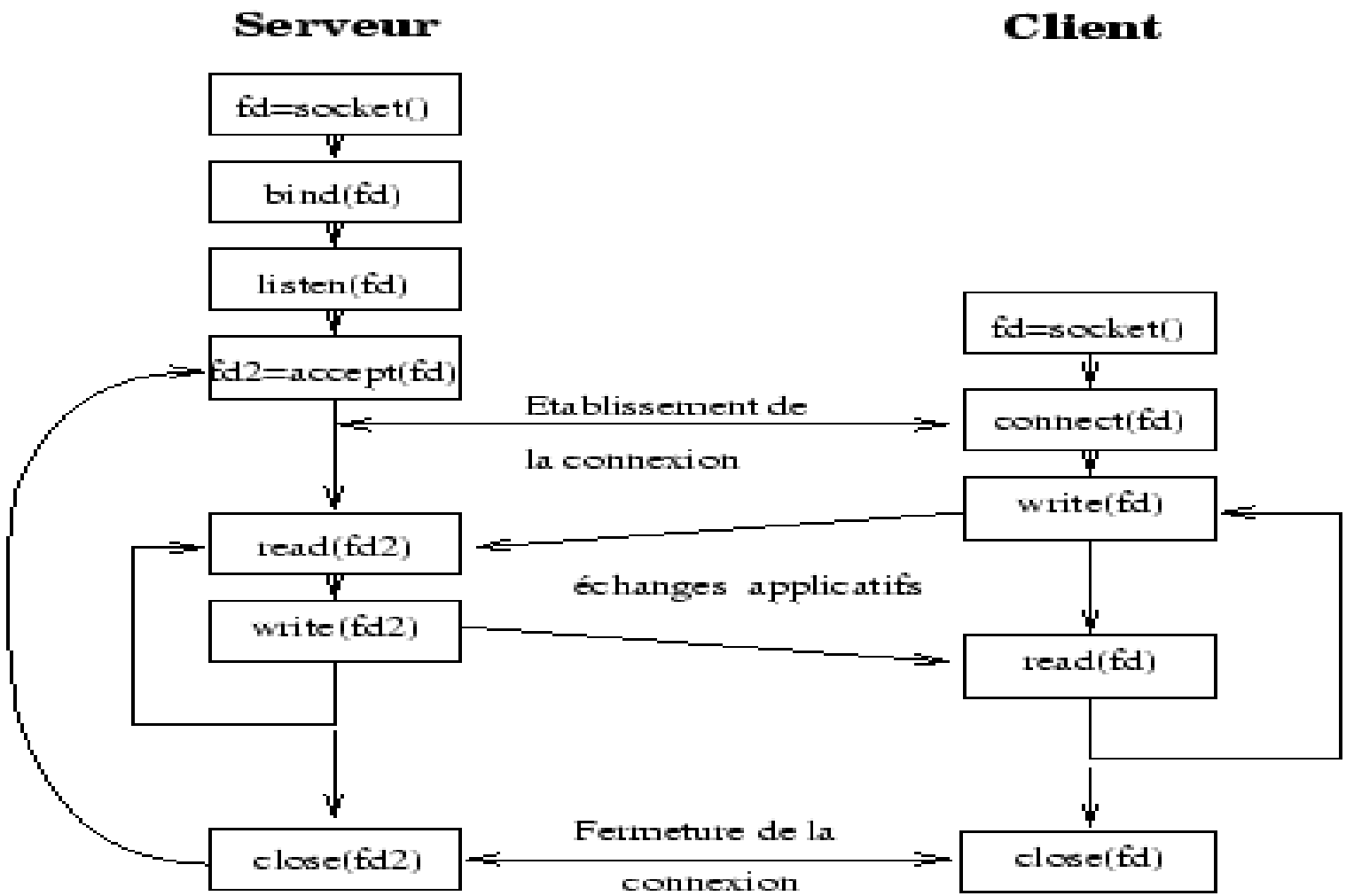
Synopsis : mode non-connecté



©F. Laissus, 2005

- Socket = une connexion réseau est vue comme un fichier ;
- Appels système en mode non-connecté (UDP) :
  - *socket* : création d'un descripteur (de fichier) ;
  - *bind* : associer À une adresse locale ;
  - *sento* : envoi de données ;
  - *recvfrom* : réception de données ;
  - *close* : libération du descripteur.

Synopsis : mode connecté



©F. Laissus, 2005

Appels système en mode connecté (TCP) :

- *socket* : création d'un descripteur (de fichier) ;
- *bind* : associer À une adresse locale ;
- *listen* : volonté d'accepter des connexion ;
- *accept* : attente jusqu'À l'arrivée d'une demande de connexion ;
- *connect* : établir la connexion ;
- *send* : envoi de données ;
- *receive* : réception de données ;
- *close* : fermeture de connexion.

Plan

1. Sockets
2. Aperçu des sockets en C
3. Sockets en Java
   - UDP
   - TCP

Programmer en Java

Installation sous linux/ubuntu :

- Java 1.6 : sudo apt-get install sun-java6-jdk
- environnement Eclipse : sudo apt-get install eclipse

Utilisation en ligne de commande :

- compilation en bytecode : javac *.java
- exécution : java uneClasse

HelloWorld.java

```
public class HelloWorld{
 public static void main( String[] args ){
   System.out.println("Hello World!");
 }}
```

Compilation & execution

```
> javac HelloWorld.java
> java HelloWorld
Hello World!
```

DatagramPacket

Cette classe permet de créer des objets qui contiendront les données d'un datagramme UDP.
Deux constructeurs sont disponibles, un pour les paquets À recevoir, l'autre pour les paquets À envoyer :

- public DatagramPacket(byte buffer[], int taille) : construit un objet pour recevoir un datagramme ; le paramètre *taille* correspond À la taille maximale des datagrammes À recevoir.
- public DatagramPacket(byte buffer[], int taille, InetAddress a, int p) : construit un objet pour envoyer un datagramme, À la machine d'adresse *a* sur le port *p*.

DatagramSocket

Cette classe permet de créer des sockets UDP pour l'envoi et la réception des datagrammes UDP.

- public DatagramSocket (int port) : crée un objet de type socket et l'attache au port UDP local passé en paramètre ;
- public void send(DatagramPacket data) : envoi le datagramme *data* ;
- public void receive(DatagramPacket data) : réception d'un datagramme, stocké dans *data*.

Client UDP en Java

Exemple de client UDP

```
InetAddress address = InetAddress.getByName("raoul.labri.fr");
int port = 4321;
String ch = "Le message"; int chl = ch.length();
byte[] message = new byte[chl]; ch.getBytes(0, chl, message, 0);

DatagramPacket dp = new DatagramPacket(message, chl, address, port);
DatagramSocket ds = new DatagramSocket();
ds.send(dp);
```

classe InetAddress

Permet de gérer les adresses IP et les noms associés.

Obtenir

- l'adresse de la machine locale : public static InetAddress getLocalHost()
- l'adresse d'une machine dont on connait le nom : public static InetAddress getByName(String hostname)
- les adresses d'une machine dont on connait le nom : public static InetAddress[] getAllByName(String hostname)

Méthodes :

- connaitre le nom de la machine : public String getHostName ()
- récupérer les 4 octets de l'adresse : public byte[] getAddress ()

Exemple :

```
System.out.println("adresse IP du serveur du LaBRI : "+InetAddress.getByName("www.labri.fr"));
```

Serveur UDP en Java

Exemple de serveur UDP

```
byte[] buffer = new byte[1024]; String ch;
DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
DatagramSocket ds = new DatagramSocket(4321);
while(true) {
  ds.receive(dp);
  ch = new String(buffer, 0, 0, dp.getLength());
  System.out.println("Paquet recu : message = " + ch +
    " - envoyeur = " + dp.getAddress().getHostName() +
    " - port = " + dp.getPort());
}
```

exercice

Exercice

écrire 2 classes `ClientUDP.java` et `ServeurUDP.java` permettant le transfert d'un message caractère par caractère.

Solution ClientUDP

```
import java.net.*;
import java.io.*;

public class ClientUDP{

public static void main( String[] args ) throws Exception{
        InetAddress address = InetAddress.getByName("localhost");
        int port = 4321;
        String ch = "Hello Bob, ici Greg, tout va bien?"; int chl = ch.length();
        byte[] message = new byte[1];
        DatagramSocket ds = new DatagramSocket();
        for (int j=0; j<1000; j++){
        long start = System.currentTimeMillis();
        for (int i=0; i<chl; i++){
                message[0]=(byte) ch.charAt(i);
                DatagramPacket dp = new DatagramPacket(message,1,address, port);
                ds.send(dp);
        }
        long end = System.currentTimeMillis();
        System.out.println("UDP: Message envoye: temps d'emission = "+(end-start)+"ms");
        }
        }

}
```

Solution ServeurUDP

```
import java.net.*;
import java.io.*;

public class ServeurUDP{

public static void main( String[] args ) throws Exception{

        byte[] buffer = new byte[1024]; String ch;
        DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
        DatagramSocket ds = new DatagramSocket(4321);
        System.out.println("Serveur UDP en attente");
        while(true) {
        ds.receive(dp);
        ch = new String(buffer, 0, 0, dp.getLength());
        System.out.print(ch);
        }
}
}
```

La classe Socket

classe Socket

La classe Socket permet de gérer une connexion TCP.

Emission/Réception de données :

- réception : public InputStream getInputStream() throws IOException
- émission : public OutputStream getOutputStream() throws IOException

Informations sur la connexion :

- adresse IP distante : public InetAddress getInetAddress()
- adresse IP locale : public InetAddress getLocalAddress()
- port distant : public int getPort()
- port local : public int getLocalPort()

La méthode close() ferme la connexion et libère les ressources du système associées À la connexion.

Exemple de client TCP

```
try {
    Socket s = new Socket("serverName",5432);
    OutputStream os = s.getOutputStream();
    InputStream  is = s.getInputStream();
    os.write((int)'a');
    System.out.println(is.read());
    s.close();
} catch(Exception e) {
    // Traitement d'erreur
}
```

Exemple de serveur TCP

```
try {
    ServerSocket ecoute = new ServerSocket(5432,5);
    Socket service = (Socket)null;
    while(true) {
        service = ecoute.accept();
        OutputStream os = service.getOutputStream();
        InputStream  is = service.getInputStream();
        os.write(is.read());
        service.close();
    }
} catch (Exception e) {
    // traitement d'erreur
}
```

Exercice : TCP

Exercice

écrire 2 classes `ClientTCP.java` et `ServeurTCP.java` permettant le transfert d'un message caractère par caractère.

Solution ClientTCP

```
import java.net.*; import java.io.*;
public class ClientTCP{
public static void main( String[] args ) throws Exception{
    String message="Hello Bob, ici Greg, tout va bien?";
    long start = System.currentTimeMillis();
    for (int i=0; i<message.length(); i++)
    {
    Socket s = new Socket("raoul.labri.fr",5432);
    OutputStream os = s.getOutputStream();
    os.write(message.charAt(i));
    s.close();
    }
    long end = System.currentTimeMillis();
    System.out.println("TCP: Message envoye: temps d'emission = "+(end-start)+"ms");
}}
```

Solution ServeurTCP

```
import java.net.*;
import java.io.*;
public class ServeurTCP{
public static void main( String[] args ) throws Exception{
    ServerSocket ecoute = new ServerSocket(5432,5);
    Socket service = (Socket)null;
    System.out.println("Serveur TCP en attente");
    while(true) {
        service = ecoute.accept();
        InputStream  is = service.getInputStream();
        System.out.print((char) is.read());
        service.close();
    }
}}
```

4 Analyse d'une trame Ethernet pas-À-pas

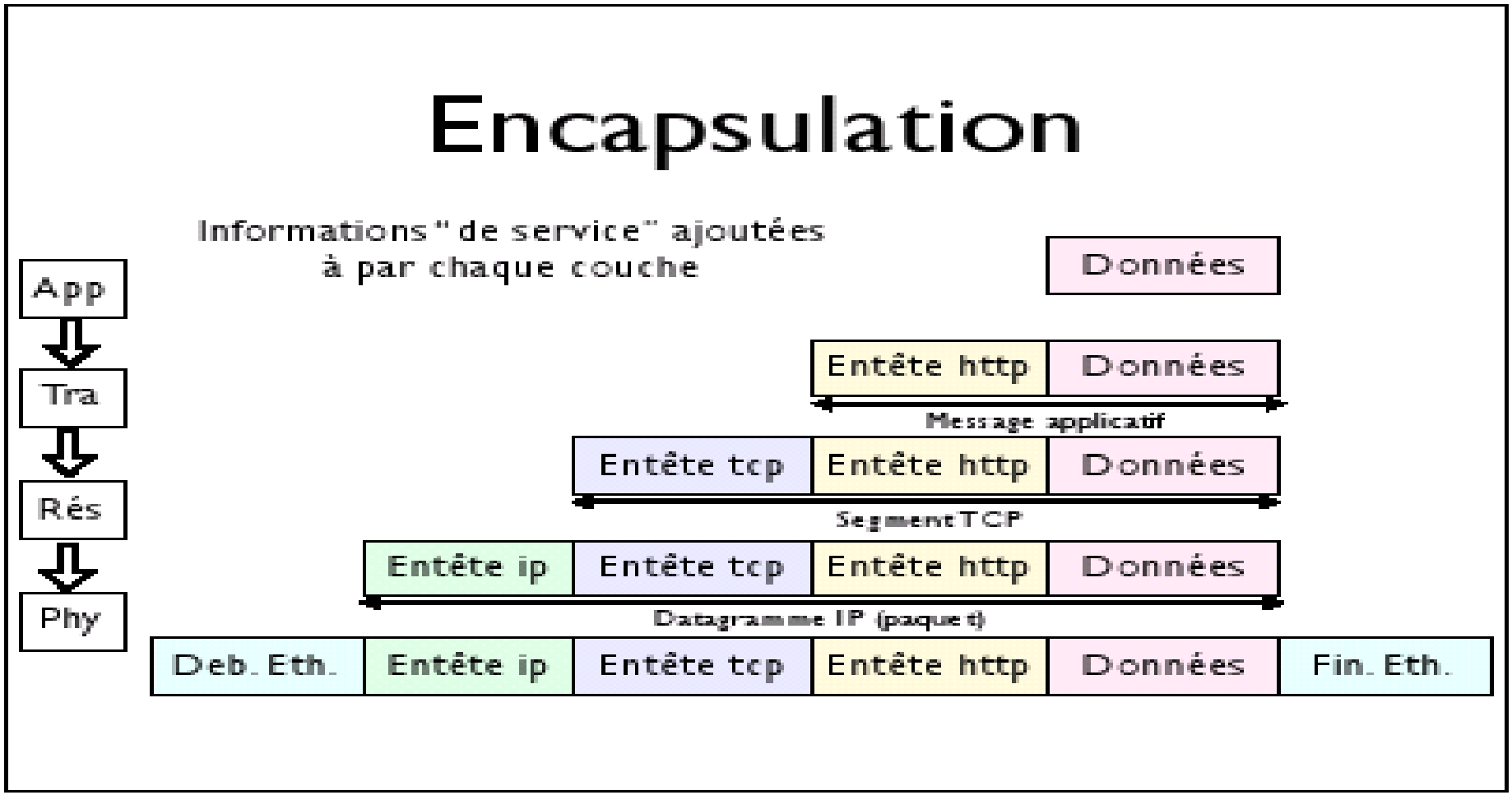
5 Avec des outils de capture

Plan

4 Analyse d'une trame Ethernet pas-À-pas

5 Avec des outils de capture

Rappel : encapsulations successives



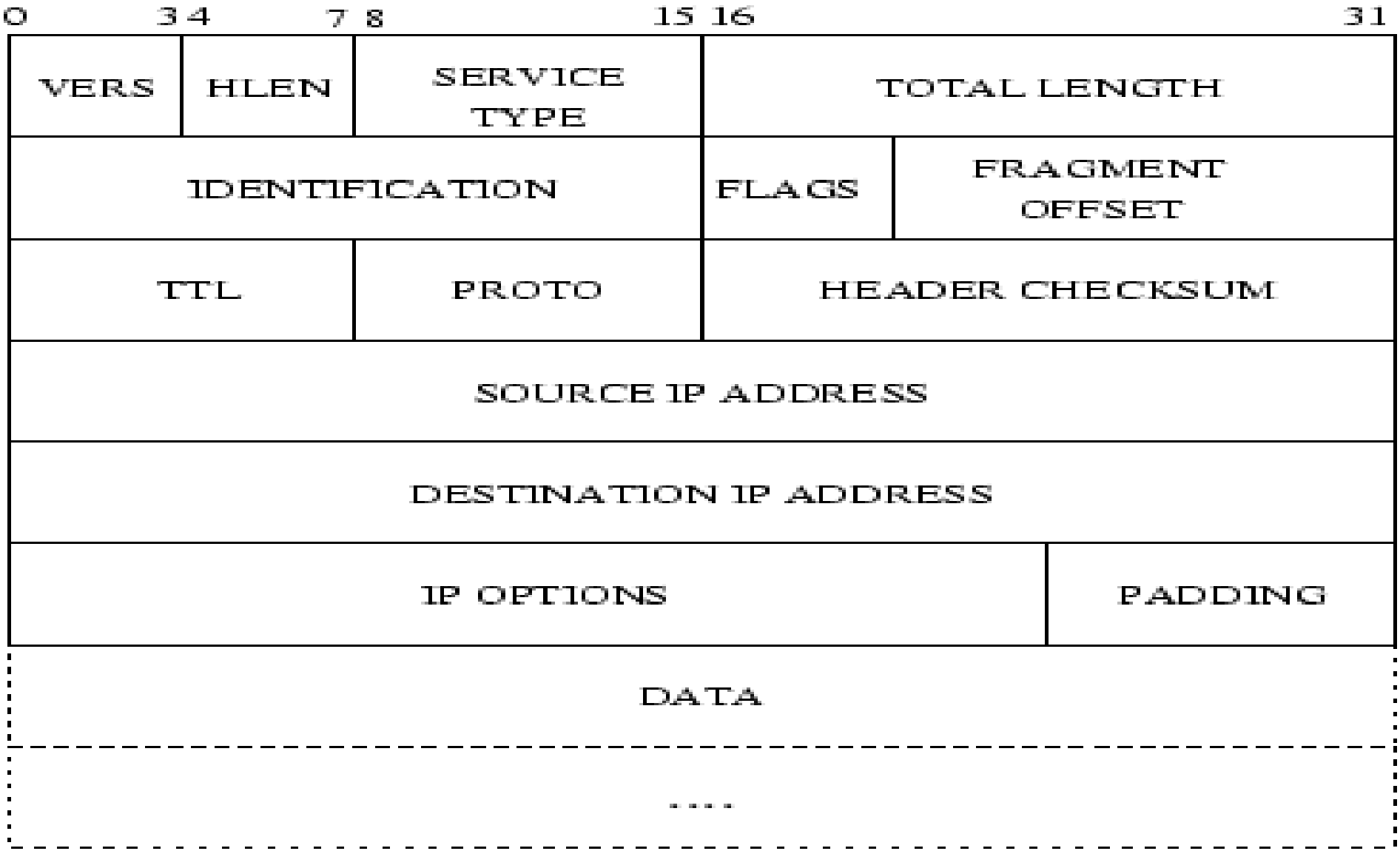
©Laissus (2004)

Structure d'une trame Ethernet

La trame de données est formée successivement de

- 8 octets : préambule
- 6 octets : adresse de destination
- 6 octets : adresse source
- 2 octets : type (en pratique) / longueur (norme)
- jusqu'À 1500 octets de données
- octets de remplissage
- 4 octets : total de contrôle

Datagramme IPv4 : rappel de la structure



Entête TCP : rappel de la structure

Transmission Control Protocol

	0 3	4 7	8 11	12 15	16 19	20 23	24 27	28 31
1	Source port				Destination port			
2	Sequence Number							
3	Acknowledgment Number							
4	Data Offset	Reserved	CWR ECE URG ACK PSH RST SYN FIN		Window			
5	Checksum				Urgent pointer			
6	Options						Padding	
7	Data							

Source : `http ://www.linux-france.org/`

Exercice : interprétation d'une trame

Interprétation d'une trame

Soit la capture d'une trame en hexadécimal suivante. Détailler les caractéristiques de la trame, en interprétant les entêtes encapsulés.

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

Indications :

- le préambule Ethernet n'a pas été capturé ;
- le type d'une trame IP est 0800 ;
- le type d'une trame TCP est 06, d'une trame UDP est 12.

Solution : interprétation de l'entête Ethernet

Entête Ethernet :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- 00 09 5b 9a c4 94 : adresse MAC destination ;
- 00 90 4b 16 5d 24 : adresse MAC source ;
- 08 00 : suite de la trame : IP/TCP

Solution : interprétation de l'entête IP

Détermination de la taille de l'entête et de la version (4 ou 6) :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- 0x45= 01000101 ;
- version : 0100 = 4 (IPv4) ;
- longueur entête : 0101 = 5 mots de 32 bits (20 octets) ;

Solution : interprétation de l'entête IP (2)

Interprétation de l'entête IPv4 : octets 2 À 4 :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- service type : 0x00 ;
- longueur totale : 0x0034 = 52 octets ;

Remarques :

- 52 octets +14 octets entete Ethernet = taille totale de la trame 66 octets ;

Solution : interprétation de l'entête IP (3)

Interprétation de l'entête IPv4 : octets 5 À 8 :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- identification : 0x68db ;
- flags (3 bits) : 0x4000 = 0100 0000 0000 0000
  - bit DF (don't fragment) = vrai ;
  - bit MF (more fragment) = faux ;
- fragment offset : 0x4000 = 0100 0000 0000 0000 = 0 : numéro 0 (pas de fragmentation) ;

Solution : interprétation de l'entête IP (4)

Interprétation de l'entête IPv4 : octets 9 À 12 :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- TTL (nombre de sauts restants) : 0x40=64 ;
- PROTO : 0x06=6=TCP ;
- Header Checksum : 0x74dd

Solution : interprétation de l'entête IP (fin)

Interprétation de l'entête IPv4 : les 8 derniers octets (13 À 20) :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- adresse IPv4 de la source : 0xc0a80002=192.168.0.2 ;
- adresse IPv4 du destinataire : 0x93d2088f=147.210.8.143.

Solution : interprétation de l'entête TCP

Interprétation de l'entête TCP : les premiers octets (de 1 À 12) :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

- port source : 0x8002 = 32770 ;
- port destination : 0x03e1 = 993 (imaps) ;
- numéro de séquence : 0xcd9cf713 = 3449616147 ;
- numéro d'acquittement : 0x496fbd4c = 1232059724.

Solution : interprétation de l'entête TCP (fin)

Interprétation de l'entête TCP : les drapeaux (octet 14) :

Adresse (hexa)	8 octets	8 octets
0000	00 09 5b 9a c4 94 00 90	4b 16 5d 24 08 00 45 00
0010	00 34 68 db 40 00 40 06	74 dd c0 a8 00 02 93 d2
0020	08 8f 80 02 03 e1 cd 9c	f7 13 49 6f bd 4c 80 10
0030	3e 96 8d 1c 00 00 01 01	08 0a 00 0e ff e0 39 d0
0040	c4 ef	

Drapeaux : 0x10 = 00010000

- URG (urgent) : faux ; ACK (acquittement) : vrai ;
- PSH (push) : faux ; RST (réinitialisation) : faux ;
- SYN (demande de connexion) : faux ; FIN (interruption) : faux.

Conclusion : la trame capturée est une trame d'acquittement du protocole TCP, entre la machine 192.168.0.2 (port 32770) et le serveur courrier 147.210.8.143 en crypté (port imaps 993).

Remarque : le serveur 147.210.8.143 est iona.labri.fr (serveur du LaBRI).

Plan

4 Analyse d'une trame Ethernet pas-À-pas

5 Avec des outils de capture

En ligne de commande

tcpdump

Utilitaire de capture de trames très complet.
Site : `www.tcpdump.org`

Exemple (extrait de Linux Magazine France 25)

```
# tcpdump -X -s 0 dst host 193.252.19.209 and port 110 and tcp

20:51:50.690745 193.253.217.180.42034 > 193.252.19.209.pop3: P 0:11(11)
     ack 79 win 5808  (DF)
0x0000   4500 003f 0000 4000 4006 c939 c1fd d9b4    E..?..@.@..9....
0x0010   c1fc 13d1 a432 006e b862 482d 520a 6b11    .....2.n.bH-R.k.
0x0020   8018 16b0 c3cf 0000 0101 080a 00a9 c278    ...............x
0x0030   0bfa 1f75 7573 6572 206f 6b6b 690d 0a      ...uuser.bidon..

20:51:50.745300 193.253.217.180.42034 > 193.252.19.209.pop3: P
    11:25(14) ack 84 win 5808  (DF)
0x0000   4500 0042 0000 4000 4006 c936 c1fd d9b4    E..B..@.@..6....
0x0010   c1fc 13d1 a432 006e b862 4838 520a 6b16    .....2.n.bH8R.k.
0x0020   8018 16b0 33d0 0000 0101 080a 00a9 c27e    ....3..........~
0x0030   0bfa 1f7a 7061 7373 2079 746b 6b76 7875    ...zpass.coucou
0x0040   0d0a
```

Qu'en pensez vous ?

Mode graphique

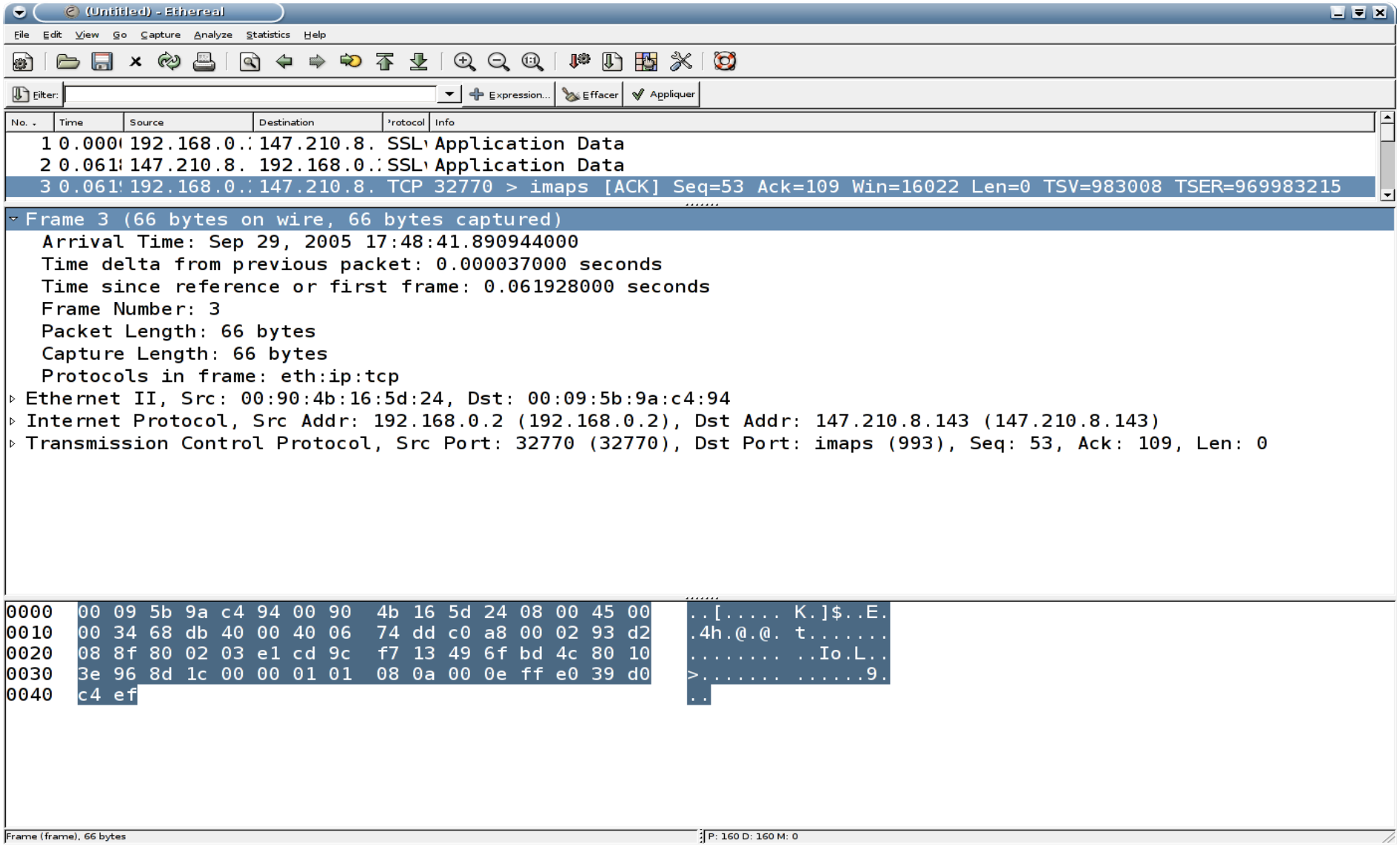
wireshark

Analyseur en mode graphique de protocoles réseaux.

- site : `www.wireshark.com`
- multi-plateforme : linux, windows

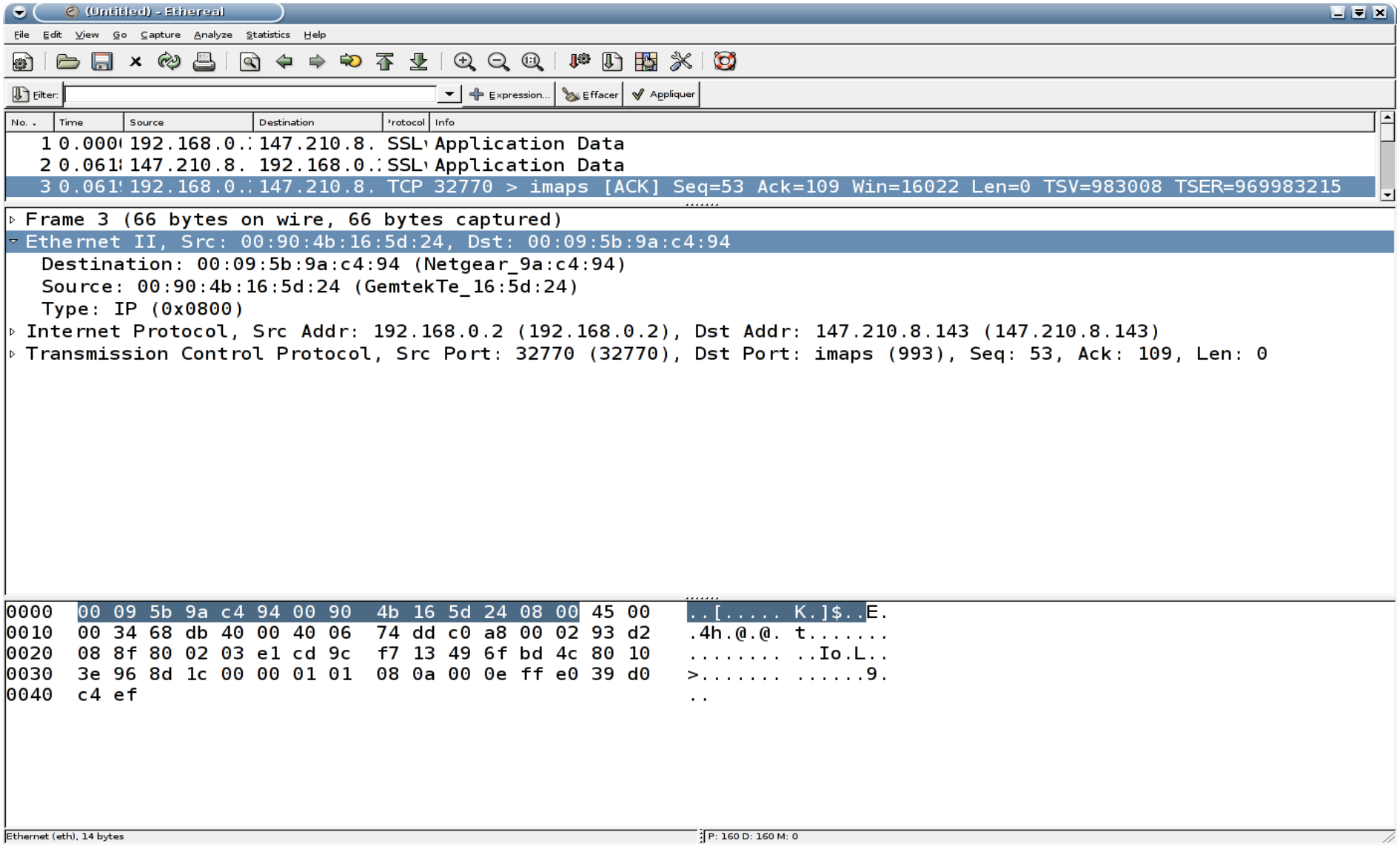
Wireshark (1)

Trame - Ethernet/IP/TCP



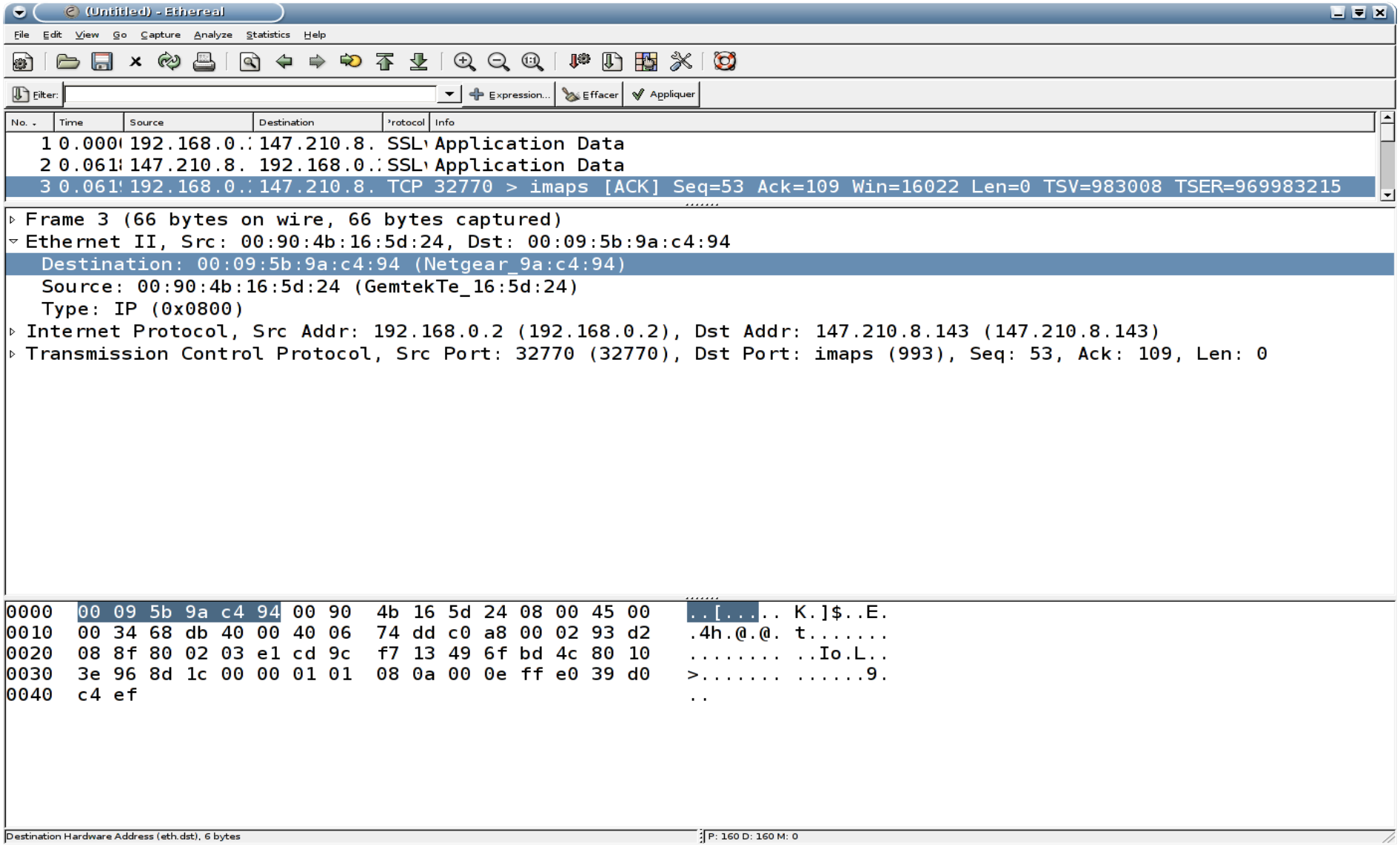
Wireshark (2)

Entête Ethernet : adresses MAC



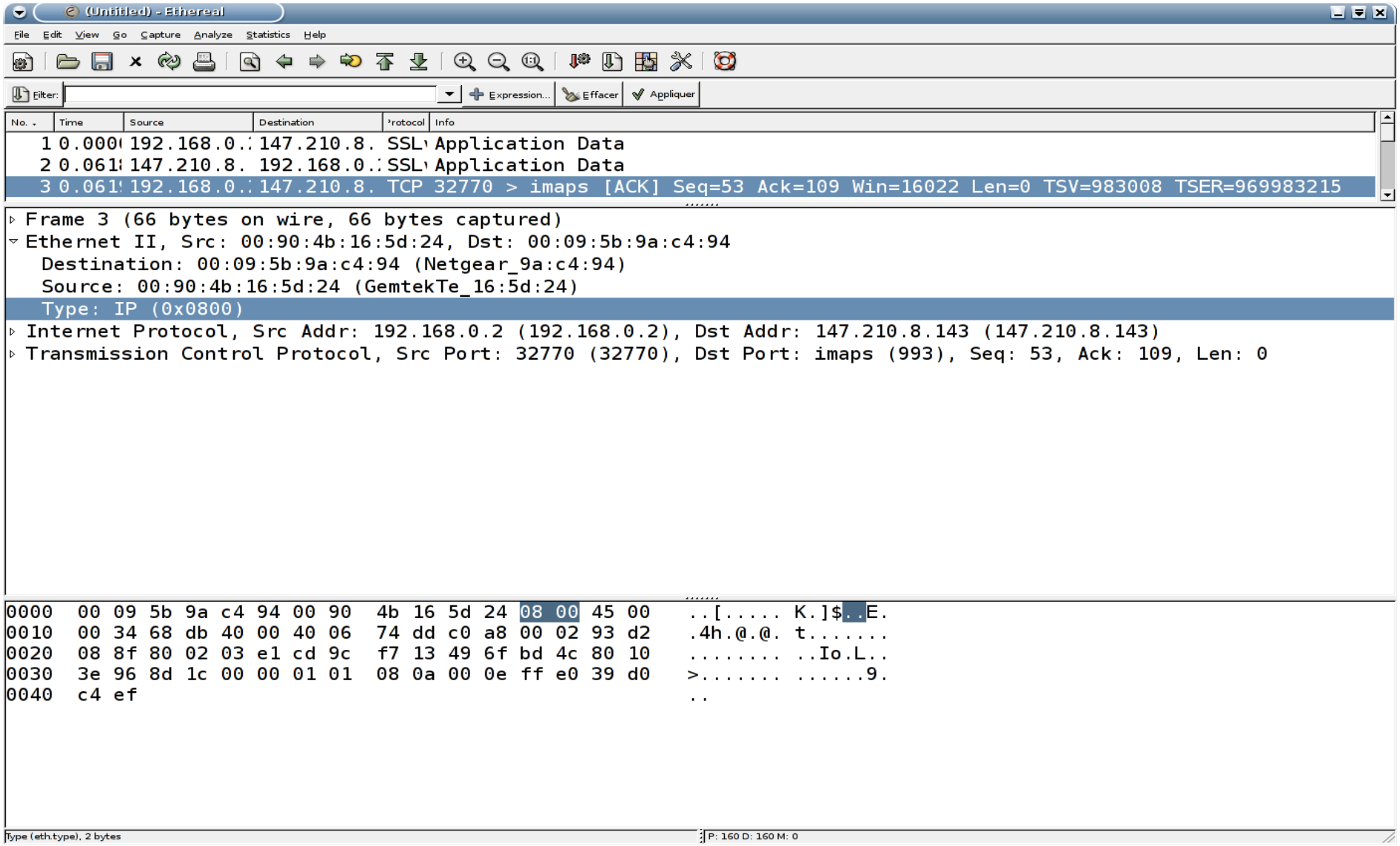
Wireshark (3)

Entête Ethernet : adresse MAC - point d'accès WIFI



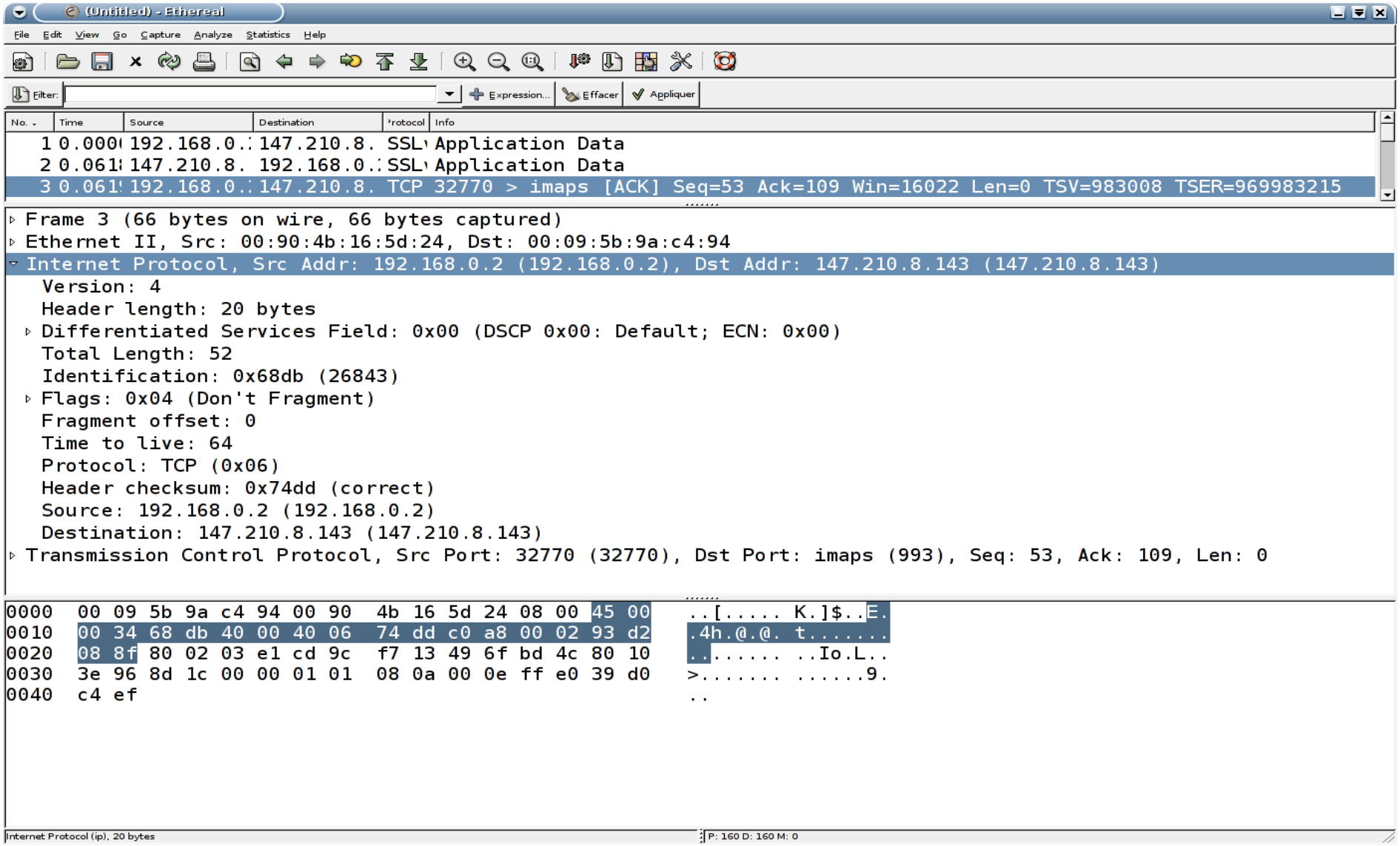
Wireshark (4)

Entête Ethernet : type de datagramme encapsulé



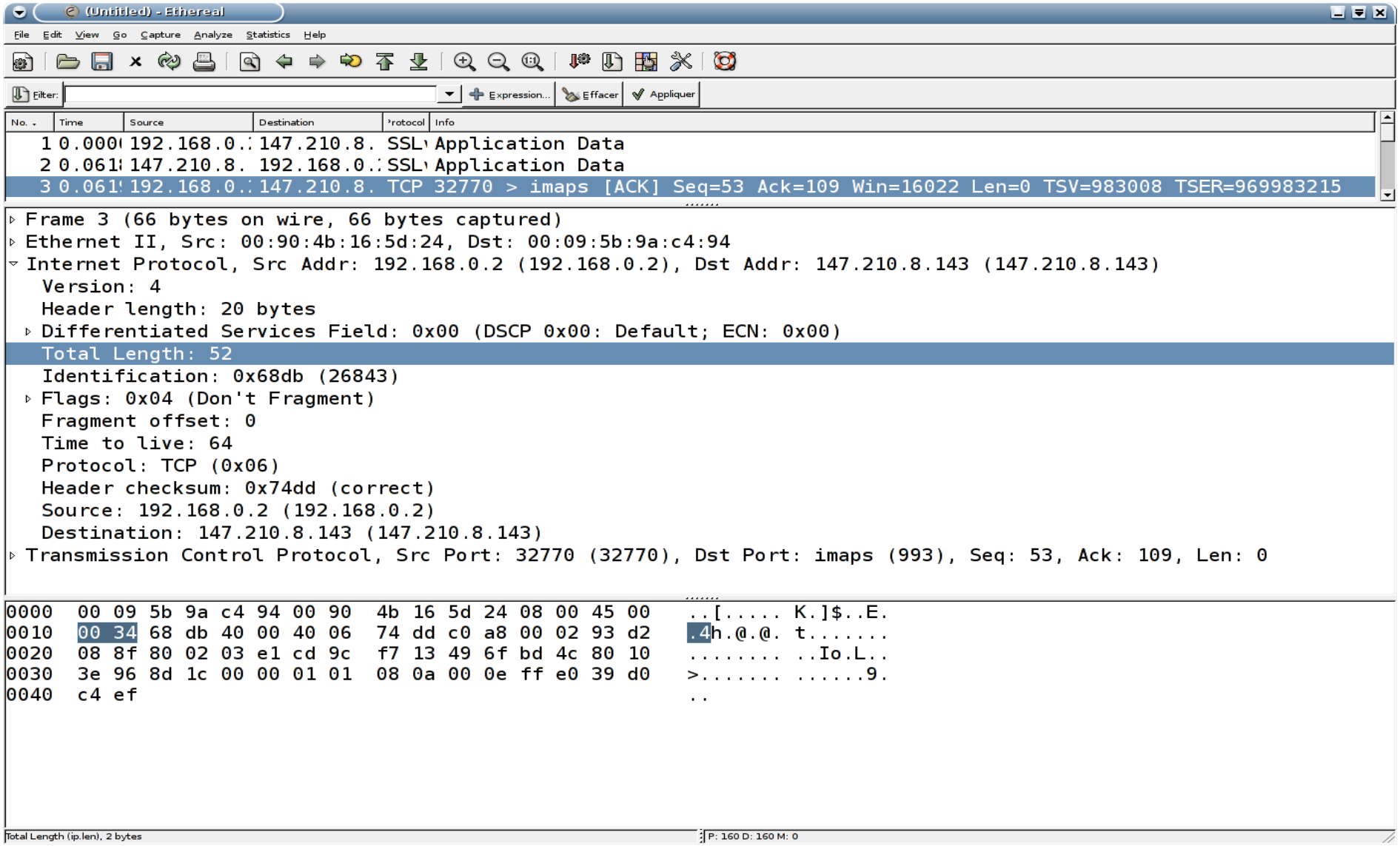
Wireshark (5)

Entête IP



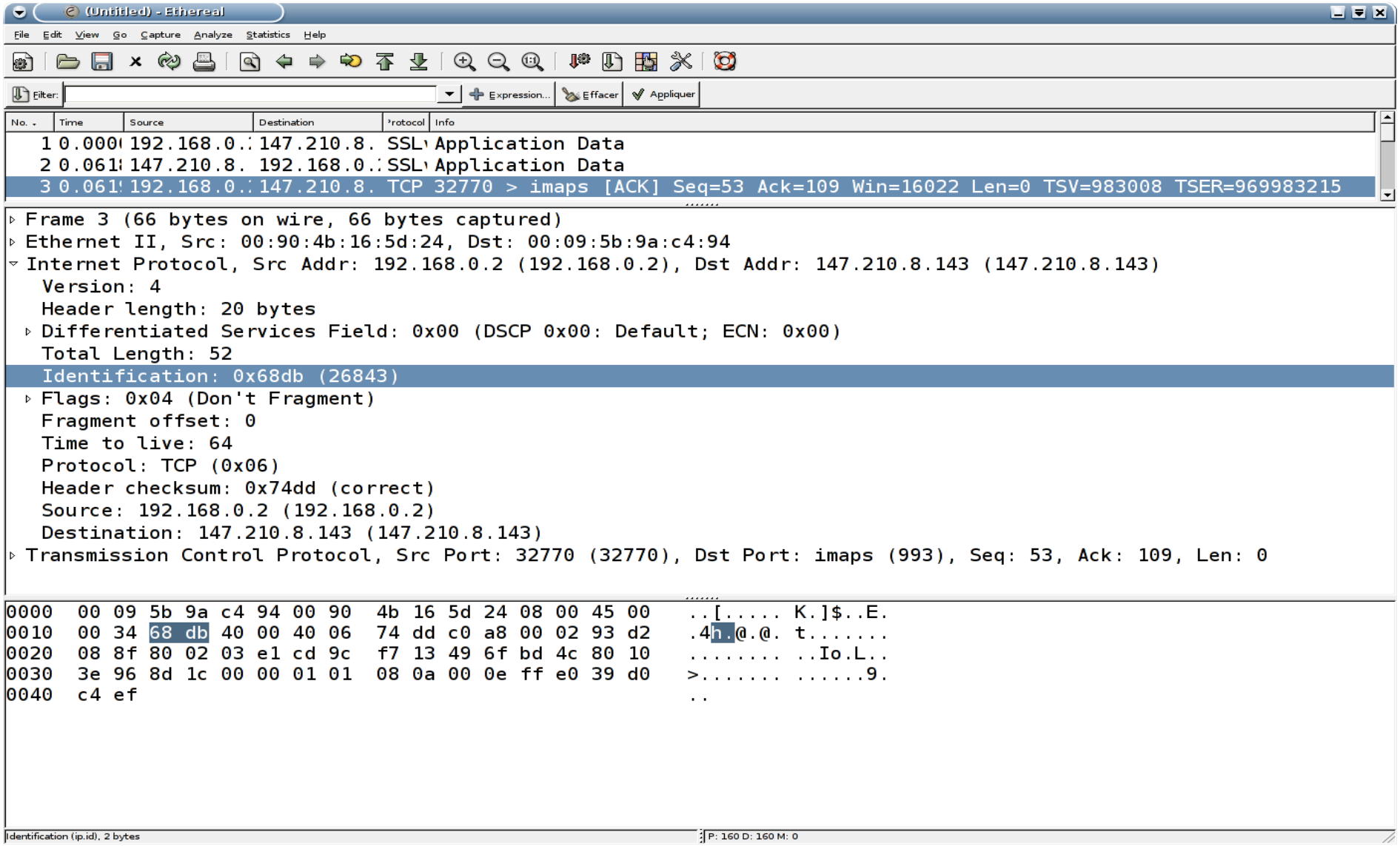
Wireshark (6)

Entête IP : champ longueur



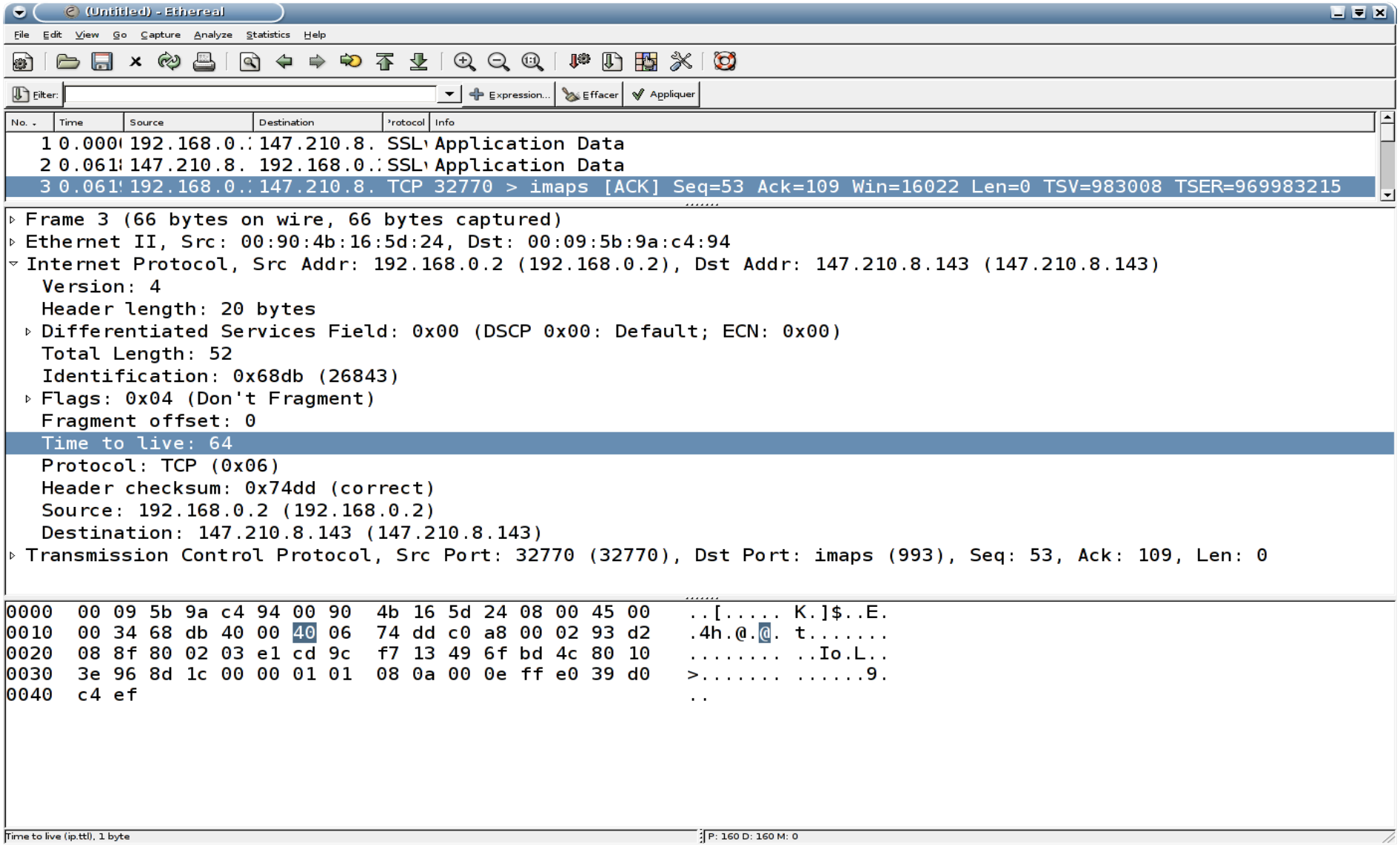
Wireshark (7)

Entête IP : champ identification



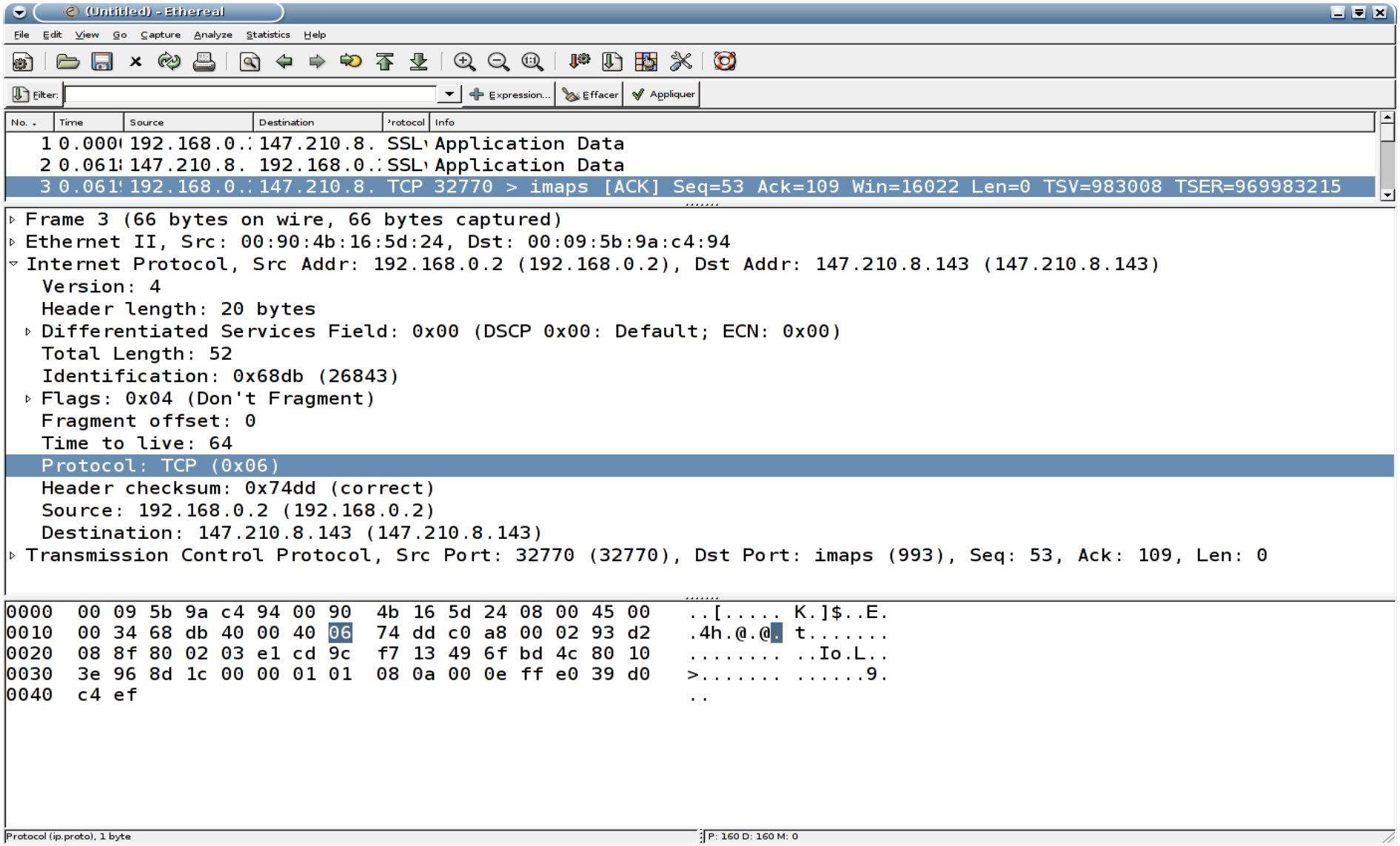
Wireshark (8)

Entête IP : champ TTL



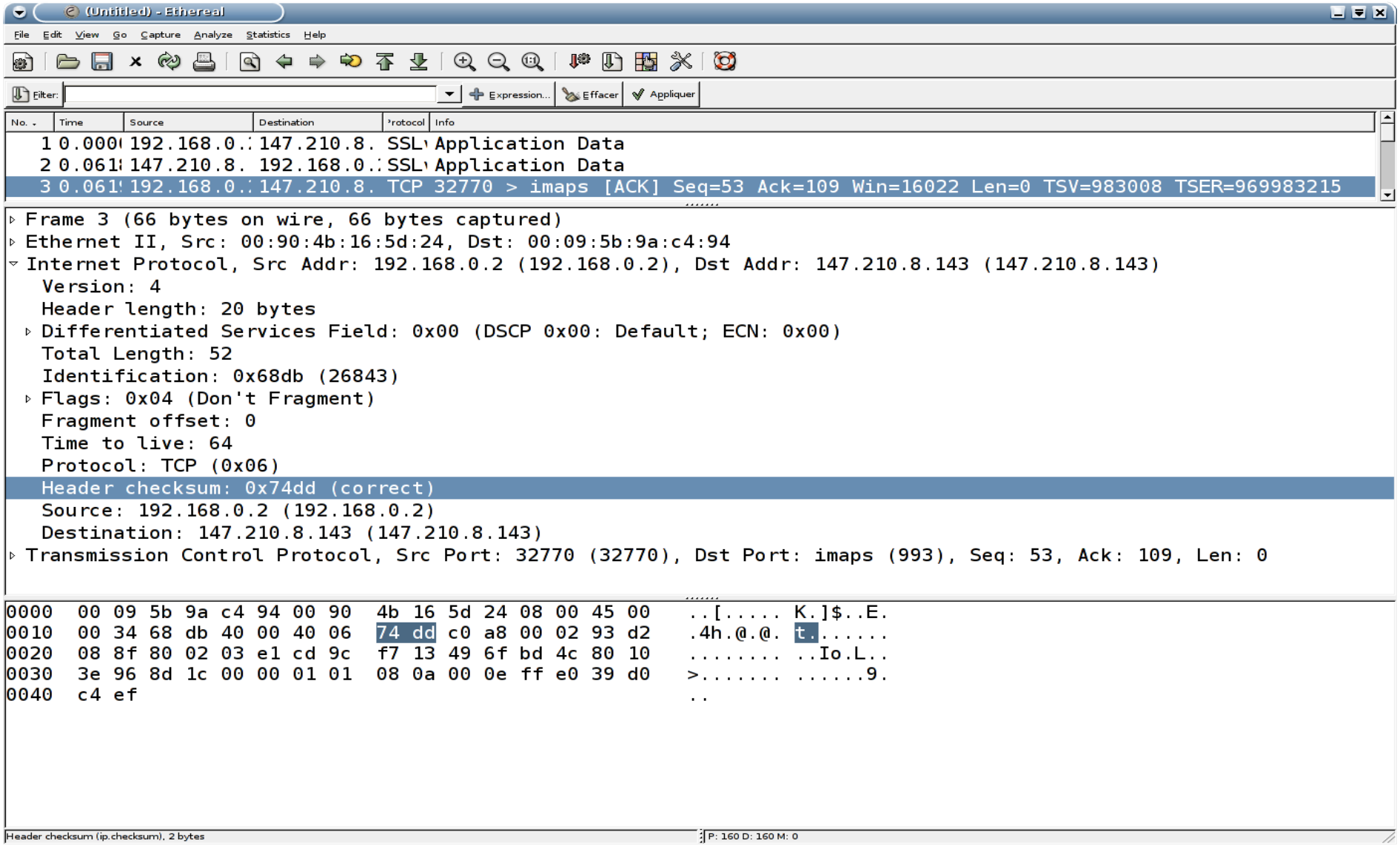
Wireshark (9)

Entête IP : type de paquet encapsulé



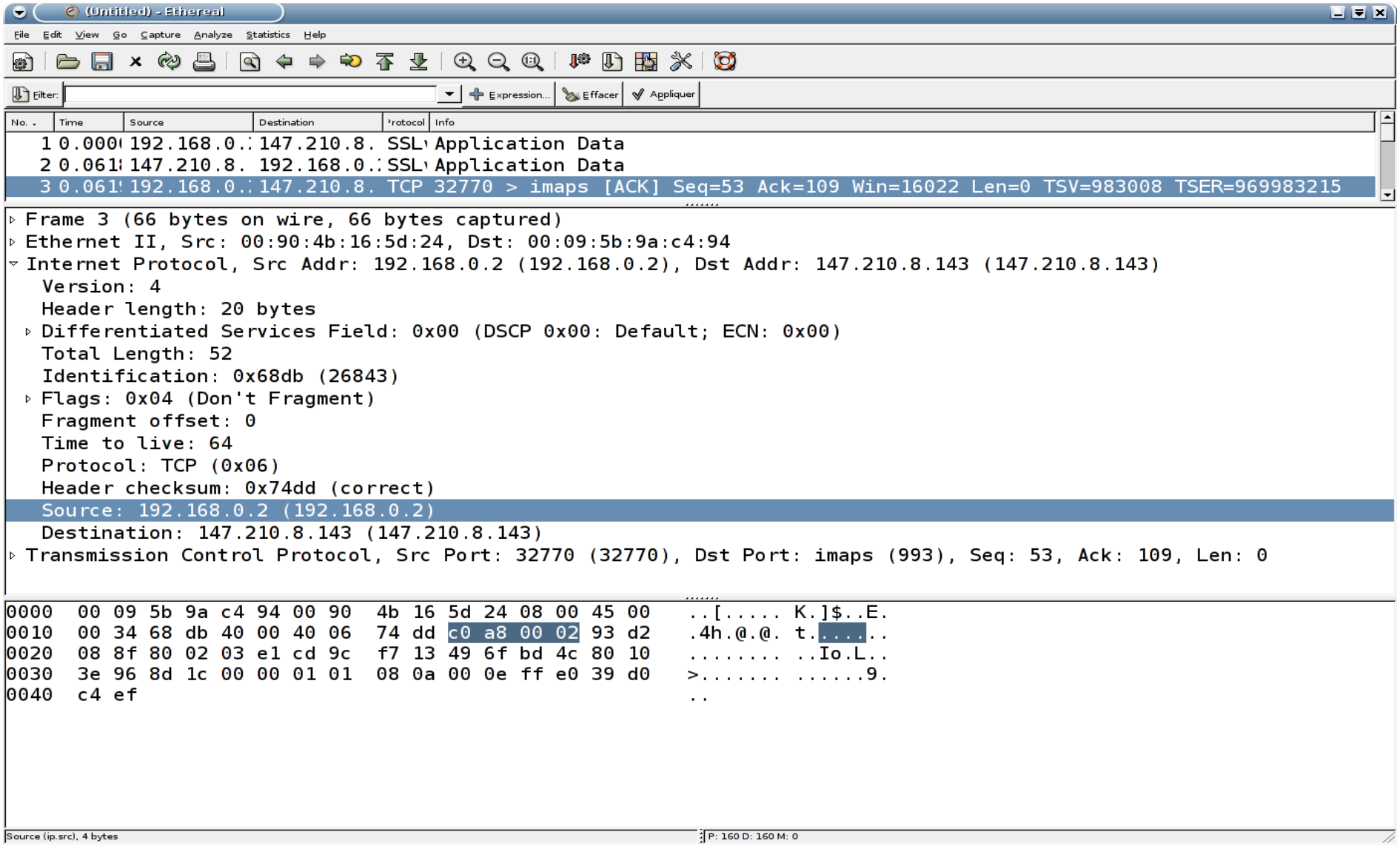
Wireshark (10)

Entête IP : somme de contrôle



Wireshark (11)

Entête IP : adresse IP source



Wireshark (12)

Paquet TCP - protocole IMAP