le cnam

Année universitaire 2018-2019

SUJET UTC505 : Réseaux et Sécurité

Examen 1e session du 26/06/2019

Responsables : *E. GRESSIER-SOUDAN, N. PIOCH*

Durée : 2 heures 30

Consignes

Calculatrice autorisée sur un équipement différent du téléphone mobile.

Tous documents autorisés

Les téléphones mobiles sont interdits
autres équipements communicants autorisés mais Internet interdit.

Les étudiants ne doivent pas communiquer entre eux.

Contrevenir à toute obligation correspond à un risque de 5 ans d'exclusion du CNAM.

Pour chaque question il est demandé une justification précise de votre réponse.
Le barème de cet examen correspond à une notation sur 21 points

Sujet de **15 pages**, celle-ci comprise.

Important : Les étudiants répondent sur les feuilles du sujet d'examen, et si nécessaire complètent leur réponse sur une copie double en faisant référence à quelle question correspond quelle réponse.

Bien mettre le numéro des questions avec leurs réponses sur la copie.

Vous rendrez une copie double au moins afin de donner les informations d'usage (nom, prénom…) et de récupérer un numéro de copie que vous ajouterez sur les feuilles du sujet d'examen. La copie doit être remise aux surveillants avec le coin cacheté.

Vous rendez le sujet rempli avec vos réponses à la fin de l'examen.

➔ *Vérifiez que vous disposez bien de la totalité des pages du sujet en début d'épreuve et signalez tout problème de reprographie le cas échéant*

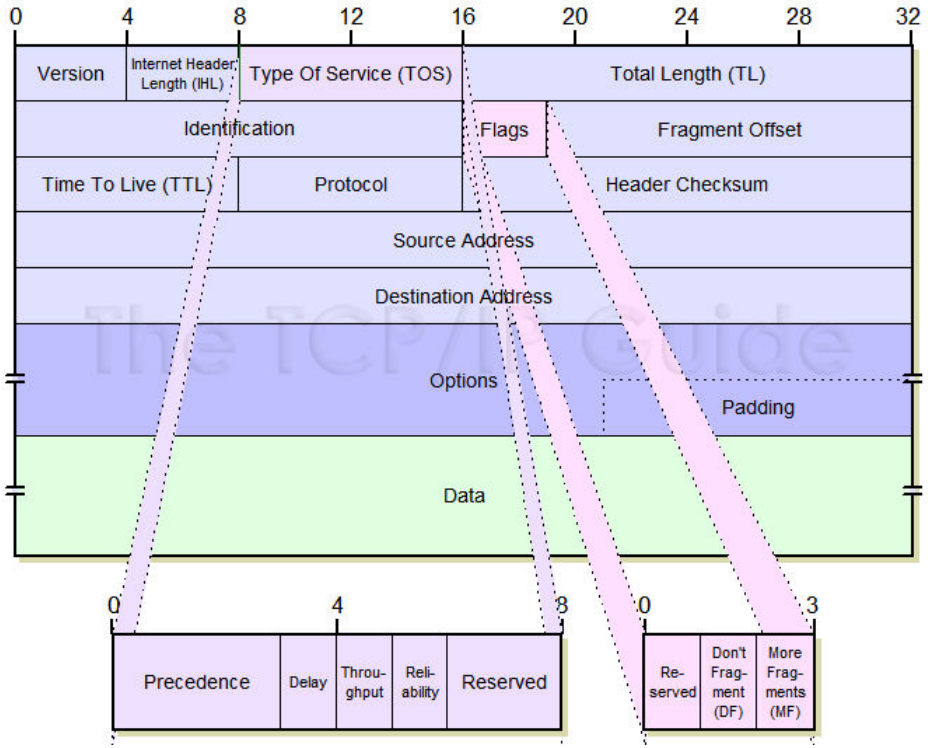
Numéro de copie :

Exercice 1 : Partie Réseaux (16 points)

On donne la structure d'une trame Ethernet :

Adresse destination	Adresse source	Type	Informations	FCS
6 octets	6 octets	2 octets	46 à 1500 octets	4 octets

On donne la structure du datagramme IP dont son entête en détail, consulté le 23 décembre 2013, Source http://www.tcpipguide.com/free/t_IPDatagramGeneralFormat.htm :



la structure d'un segment TCP dont l'entête en détail, consulté le 23 décembre 2013 source http://caleudum.wordpress.com/2011/05/08/tcp-header-format/ :

TCP Header

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number (if ACK set)																															
96	Data offset				Reserved				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
128	Checksum																Urgent pointer (if URG set)															
160 …	Options (if Data Offset > 5) …																								padding							

Les indicateurs qui nous intéressent sont :

- URG : Signale la présence de données **urg**entes
- ACK : signale que le segment contient un accusé de réception (**ack**nowledgement)
- PSH : données à envoyer et délivrer tout de suite (**p**u**sh**)
- RST : rupture anormale de la connexion (re**s**e**t**)
- SYN : demande de **syn**chronisation ou établissement de connexion
- FIN : demande la **fin** de la connexion

On s'intéresse à une trace Wireshark qui formalise un échange client/serveur. Elle vous est donnée ci-dessous.

1	0.000000	10.20.144.150	10.20.144.151	TCP	74	35974 → 21 [SYN] Seq=0 Win=32648 Len=0 MSS=1380 WS=1 TSval=1657560000 TSecr=0
2	0.000320	10.20.144.151	10.20.144.150	TCP	78	21 → 35974 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1356 WS=1 TSval=1657390000 TSecr=1
3	0.000570	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [ACK] Seq=1 Ack=1 Win=32648 Len=0 TSval=1657560000 TSecr=1657390000
4	0.060630	10.20.144.151	10.20.144.150	FTP	106	Response: 220-QTCP at fran.csg.stercomm.com.
5	0.275440	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [ACK] Seq=1 Ack=37 Win=32648 Len=0 TSval=1657560500 TSecr=1657390000
6	0.275760	10.20.144.151	10.20.144.150	FTP	126	Response: 220 Connection will close if idle more than 5 minutes.
7	0.276140	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [ACK] Seq=1 Ack=93 Win=32648 Len=0 TSval=1657560500 TSecr=1657390000
8	4.216600	10.20.144.150	10.20.144.151	FTP	81	Request: USER cdts3500
9	4.217350	10.20.144.151	10.20.144.150	FTP	91	Response: 331 Enter password.
10	4.217630	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=16 Ack=114 Win=32648 Len=0 TSval=1657564500 TSecr=1657394000
11	7.639420	10.20.144.150	10.20.144.151	FTP	81	Request: PASS cdts3500
12	7.843260	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=114 Ack=31 Win=16384 Len=0 TSval=1657397500 TSecr=1657568000
13	8.184000	10.20.144.151	10.20.144.150	FTP	95	Response: 230 CDTS3500 logged on.
14	8.184360	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=31 Ack=139 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
15	8.185040	10.20.144.150	10.20.144.151	FTP	72	Request: SYST
16	8.185260	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=139 Ack=37 Win=16384 Len=0 TSval=1657398000 TSecr=1657568500
17	8.192750	10.20.144.151	10.20.144.150	FTP	147	Response: 215 OS/400 is the remote operating system. The TCP/IP version is "V5R2M0".
18	8.193000	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=37 Ack=216 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
19	8.193570	10.20.144.150	10.20.144.151	FTP	80	Request: SITE NAMEFMT
20	8.193780	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=216 Ack=51 Win=16384 Len=0 TSval=1657398000 TSecr=1657568500
21	8.194900	10.20.144.151	10.20.144.150	FTP	105	Response: 250 Now using naming format "0".
22	8.195140	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=51 Ack=251 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
23	8.195700	10.20.144.150	10.20.144.151	FTP	71	Request: PWD
24	8.195910	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=251 Ack=56 Win=16384 Len=0 TSval=1657398000 TSecr=1657568500
25	8.197050	10.20.144.151	10.20.144.150	FTP	106	Response: 257 "CDTS3500" is current library.
26	8.197280	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=56 Ack=287 Win=32648 Len=0 TSval=1657568500 TSecr=1657398000
27	20.765720	10.20.144.150	10.20.144.151	FTP	72	Request: PASV
28	20.766000	10.20.144.151	10.20.144.150	TCP	70	21 → 35974 [PSH, ACK] Seq=287 Ack=62 Win=16384 Len=0 TSval=1657410500 TSecr=1657581000
29	20.787770	10.20.144.151	10.20.144.150	FTP	121	Response: 227 Entering Passive Mode (10,20,144,151,62,141).
30	20.788010	10.20.144.150	10.20.144.151	TCP	66	35974 → 21 [PSH, ACK] Seq=62 Ack=338 Win=32648 Len=0 TSval=1657581000 TSecr=1657410500

La trace ne tenant pas sur une seule page, la suite vous est donnée page suivante, seule la trame 30 est commune aux deux images, c'est juste pour faire le lien.

```
30 20.788010   10.20.144.150   10.20.144.151   TCP      66 35974 → 21 [PSH, ACK] Seq=62 Ack=338 Win=32648 Len=0 TSval=1657581000 TSecr=1657410500
31 20.797560   10.20.144.150   10.20.144.151   TCP      74 35976 → 16013 [SYN] Seq=0 Win=32768 Len=0 MSS=1380 WS=1 TSval=1657581000 TSecr=0
32 20.797850   10.20.144.151   10.20.144.150   TCP      78 16013 → 35976 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1356 WS=1 TSval=1657410500 TSecr=1657581000
33 20.798130   10.20.144.150   10.20.144.151   TCP      66 35976 → 16013 [ACK] Seq=1 Ack=1 Win=32768 Len=0 TSval=1657581000 TSecr=1657410500
34 20.798250   10.20.144.150   10.20.144.151   FTP      91 Request: RETR qgpl/apkeyf.apkeyf
35 20.798450   10.20.144.151   10.20.144.150   TCP      70 21 → 35974 [PSH, ACK] Seq=338 Ack=87 Win=16384 Len=0 TSval=1657410500 TSecr=1657581000
36 21.202190   10.20.144.151   10.20.144.150   FTP     132 Response: 150 Retrieving member APKEYF in file APKEYF in library QGPL.
37 21.202460   10.20.144.150   10.20.144.151   TCP      66 35974 → 21 [PSH, ACK] Seq=87 Ack=400 Win=32648 Len=0 TSval=1657581500 TSecr=1657411000
38 21.313290   10.20.144.151   10.20.144.150   FTP-DA…  509 FTP Data: 439 bytes (PASV) (RETR qgpl/apkeyf.apkeyf)
39 21.393980   10.20.144.151   10.20.144.150   TCP      70 16013 → 35976 [FIN, PSH, ACK] Seq=440 Ack=1 Win=32768 Len=0 TSval=1657411500 TSecr=1657581000
40 21.394160   10.20.144.151   10.20.144.150   FTP     113 Response: 250 File transfer completed successfully.
41 21.394310   10.20.144.150   10.20.144.151   TCP      66 35976 → 16013 [ACK] Seq=1 Ack=441 Win=32768 Len=0 TSval=1657581500 TSecr=1657411500
42 21.394430   10.20.144.150   10.20.144.151   TCP      66 35974 → 21 [PSH, ACK] Seq=87 Ack=443 Win=32648 Len=0 TSval=1657581500 TSecr=1657411500
43 22.169470   10.20.144.150   10.20.144.151   TCP      66 35976 → 16013 [FIN, PSH, ACK] Seq=1 Ack=441 Win=32768 Len=0 TSval=1657582500 TSecr=1657411500
44 22.169800   10.20.144.151   10.20.144.150   TCP      70 16013 → 35976 [PSH, ACK] Seq=441 Ack=2 Win=32768 Len=0 TSval=1657412000 TSecr=1657582500
45 31.007220   10.20.144.150   10.20.144.151   FTP      72 Request: QUIT
46 31.007560   10.20.144.151   10.20.144.150   TCP      70 21 → 35974 [PSH, ACK] Seq=443 Ack=93 Win=16384 Len=0 TSval=1657421000 TSecr=1657591500
47 31.007750   10.20.144.151   10.20.144.150   FTP     101 Response: 221 QUIT subcommand received.
48 31.007830   10.20.144.151   10.20.144.150   TCP      70 21 → 35974 [FIN, PSH, ACK] Seq=474 Ack=93 Win=16384 Len=0 TSval=1657421000 TSecr=1657591500
49 31.008000   10.20.144.150   10.20.144.151   TCP      66 35974 → 21 [PSH, ACK] Seq=93 Ack=474 Win=32648 Len=0 TSval=1657591500 TSecr=1657421000
50 31.008810   10.20.144.150   10.20.144.151   TCP      66 35974 → 21 [FIN, PSH, ACK] Seq=93 Ack=474 Win=32648 Len=0 TSval=1657591500 TSecr=1657421000
51 31.008840   10.20.144.150   10.20.144.151   TCP      66 35974 → 21 [PSH, ACK] Seq=94 Ack=475 Win=32648 Len=0 TSval=1657591500 TSecr=1657421000
52 31.009050   10.20.144.151   10.20.144.150   TCP      70 21 → 35974 [PSH, ACK] Seq=475 Ack=94 Win=16384 Len=0 TSval=1657421000 TSecr=1657591500
```

Pour vous aider, on donne aussi une vue très résumée des échanges sur les différentes connexions :

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.20.144.150	35974	10.20.144.151	21	44	3569	23	1618	21	1951	0.000000	31.0090	417	
10.20.144.150	35976	10.20.144.151	16013	8	999	4	272	4	727	20.797560	1.3722	1585	

Question 1 : Dans la trace deux ouvertures de connections TCP peuvent être observées ? Donner le numéro de la trame, pour chaque connexion où la demande d'ouverture de connexion est initialisée. Comment reconnaissez vous que ce sont des ouvertures de connexion ? Ces connexions sont-elles complètes, c'est-à-dire sont-elles bien fermées dans la trace ? **(2 points)**

Correction :

En ligne 1, un premier "three way handshake" est visible à l'aide de l'échange SYN/SYN-ACK/ACK qui s'appuie aussi sur les lignes 2 et 3.

En ligne 31, un second "three way handshake" est visible à l'aide de l'échange SYN/SYN-ACK/ACK qui s'appuie aussi sur les lignes 32 et 33.

Les deux connexions se ferment dans la trace : la première connexion ouverte se ferme lignes 48 et 49 avec un FIN/ACK puis en 50 et 51 avec un FIN/ACK en sens inverse, la seconde connexion ouverte se ferme ligne 39 et 41 avec un FIN/ACK puis en 43 et 44 avec un FIN/ACK en sens inverse.

Question 2 : Pour chaque connexion dans la trace, quelle est l'adresse IP et le numéro de port de l'application qui initie la connexion ? **(1 point)**
Correction :
Connexion 1 : 10.20.144.150, port 35974
Connexion 2 : 10.20.144.150, port 35976
Ce sont les extrémités qui activent la connexion, donc correspondent au rôle actif comme présenté dans le cours.

Question 3 : Pour chaque connexion dans la trace, quelle est l'adresse IP et le numéro de port de l'application qui accepte l'ouverture de connexion ? **(1 point)**
Correction :
Connexion 1 : 10.20.144.151, port 21
Connexion 2 : 10.20.144.151, port 16013
Ce sont les extrémités qui acceptent l'ouverture de la connexion, donc correspondent au rôle passif comme présenté dans le cours.

On s'intéresse en particulier à la trame 31.

No.	Time	Source	Destination	Protocol	Length	Info
31	20.797560	10.20.144.150	10.20.144.151	TCP	74	35976 → 16013 [SYN] Seq=0 Win=32768 Len=0 MSS=1380 WS=1 TSval=1657581000 TSecr=0
32	20.797850	10.20.144.151	10.20.144.150	TCP	78	16013 → 35976 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1356 WS=1 TSval=1657410500 TSecr=1657581000
33	20.798130	10.20.144.150	10.20.144.151	TCP	66	35976 → 16013 [ACK] Seq=1 Ack=1 Win=32768 Len=0 TSval=1657581000 TSecr=1657410500

```
> Frame 31: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
v Ethernet II, Src: IbmRisc6_9c:14:fe (00:06:29:9c:14:fe), Dst: IbmRisc6_9c:14:ae (00:06:29:9c:14:ae)
    > Destination: IbmRisc6_9c:14:ae (00:06:29:9c:14:ae)
    > Source: IbmRisc6_9c:14:fe (00:06:29:9c:14:fe)
      Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 10.20.144.150, Dst: 10.20.144.151
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 60
      Identification: 0x2d80 (11648)
    > Flags: 0x4000, Don't fragment
      Time to live: 64
      Protocol: TCP (6)
      Header checksum: 0xd7e6 [validation disabled]
      [Header checksum status: Unverified]
      Source: 10.20.144.150
      Destination: 10.20.144.151
v Transmission Control Protocol, Src Port: 35976, Dst Port: 16013, Seq: 0, Len: 0
      Source Port: 35976
      Destination Port: 16013
      [Stream index: 1]
      [TCP Segment Len: 0]
      Sequence number: 0    (relative sequence number)
      [Next sequence number: 0    (relative sequence number)]
      Acknowledgment number: 0
      1010 .... = Header Length: 40 bytes (10)
    > Flags: 0x002 (SYN)
      Window size value: 32768
      [Calculated window size: 32768]
      Checksum: 0x2c86 [unverified]
```

```
0000  00 06 29 9c 14 ae 00 06  29 9c 14 fe 08 00 45 00   ··)····· )·····E·
0010  00 3c 2d 80 40 00 40 06  d7 e6 0a 14 90 96 0a 14   ·<-·@·@· ········
0020  90 97 8c 88 3e 8d 01 f9  8b d8 00 00 00 00 a0 02   ····>··· ········
0030  80 00 2c 86 00 00 02 04  05 64 01 03 03 00 01 01   ··,····· ·d······
0040  08 0a 62 cc ad c8 00 00  00 00                     ··b····· ··
```

➔ *Vérifiez que vous disposez bien de la totalité des pages du sujet en début d'épreuve et signalez tout problème de reprographie le cas échéant*

Question 4 : Analyse de trame **(4 points)**

- Délimiter l'entête de la trame Ethernet dans la capture en hexadécimal ci-dessous. **(0,25 point)**
- Délimiter l'entête du datagramme IP dans la capture en hexadécimal ci-dessous. **(0,25 point)**
- Délimiter l'entête du segment TCP dans la capture en hexadécimal ci-dessous. **(0,25 point)**

Ne pas hésiter à utiliser des couleurs différentes pour que votre réponse soit facile à lire.

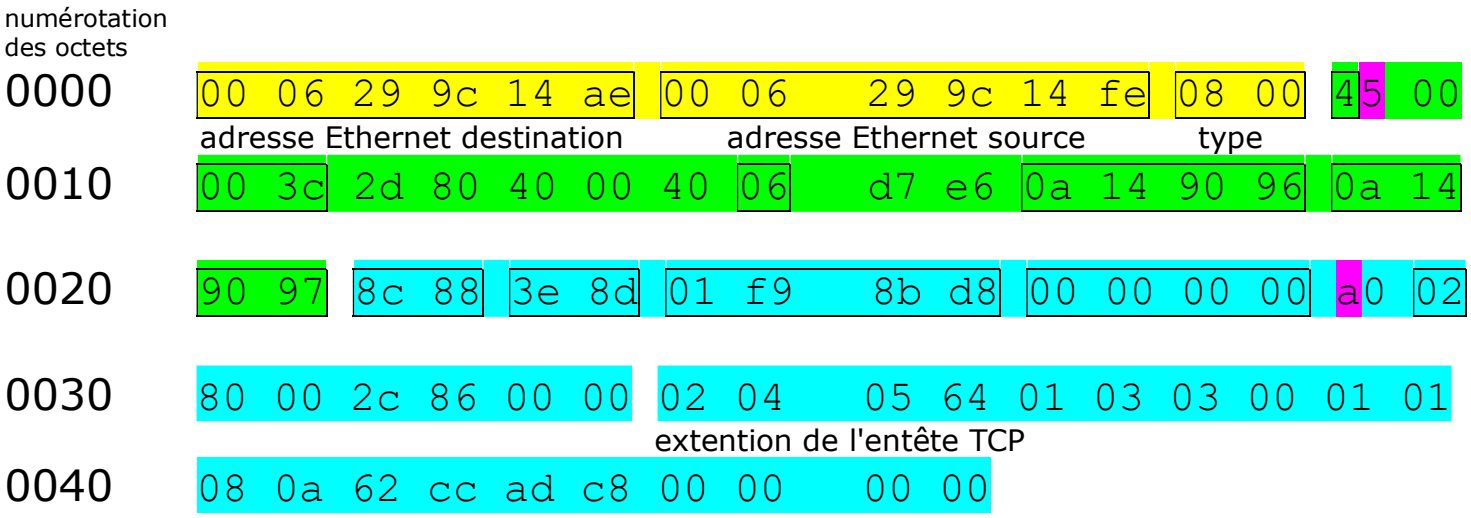
```
0000    00 06 29 9c 14 ae 00 06   29 9c 14 fe 08 00 45 00

0010    00 3c 2d 80 40 00 40 06   d7 e6 0a 14 90 96 0a 14

0020    90 97 8c 88 3e 8d 01 f9   8b d8 00 00 00 00 a0 02

0030    80 00 2c 86 00 00 02 04   05 64 01 03 03 00 01 01

0040    08 0a 62 cc ad c8 00 00   00 00
```

Attention la colonne la plus à gauche numérote les lignes et cette numérotation est hexadécimale.

Retrouver les champs suivants dans la trace hexadécimale ci-dessus :

- Quelle est l'adresse Ethernet destination en **hexadécimal** ? **(0,25 point)**
- Quelle est l'adresse Ethernet source en **hexadécimal** ? **(0,25 point)**
- Quel est le type de la trame en **hexadécimal** ? **(0,25 point)**
- Quelle est la version du protocole IP en **décimal** ? **(0,25 point)**
- Quelle est la longueur de l'entête IP en **décimal** ? **(0,25 point)**
- Quelle est l'adresse IP source en **hexadécimal** ? **(0,25 point)**
- Quelle est l'adresse IP destination en **hexadécimal** ? **(0,25 point)**
- Quel est le numéro du protocole indiqué par l'entête IP et transporté dans la charge utile du datagramme IP en **héxadécimal**, c'est TCP ou UDP ? **(0,25 point)**
- Quelle est la longueur totale du datagramme **en décimal** ? **(0,25 point)**
- Quel est le numéro de port source en **héxadécimal** ? **(0,25 point**)
- Quel est le numéro de port destination en **héxadécimal** ? **(0,25 point)**
- Quel est la valeur du numéro de séquence absolu dans l'entête TCP en **hexadécimal** ? **(0,25 point)**
- Quel est le seul "flag" positionné dans l'entête TCP ? **(0,25 point)**

Correction :



entête Ethernet, entête IP, entête TCP

L'évaluation de la taille de l'entête est un peu compliquée. Il faut regarder le champ data offset, s'il est supérieur à 5 l'entête est supérieure à 20 octets qui est la taille la plus courante. Ici il vaut a > 5 donc l'entête ne sera pas de 20 octets. a correspond à 10 (sa valeur max est 15, ou encore f en héxadécimal), il faut le multiplier par 4 pour avoir la taille de l'entête : 4*10 = 40, l'entête fait donc 40 octets, les 20 octets en plus correspondent à des options. Sinon, on peut s'en sortir autrement et lire la taille de l'entête en direct dans la partie explicative de la fenêtre Wireshark.

- Quelle est l'adresse Ethernet destination en **hexadécimal** : 00 06 29 9c 14 ae
- Quelle est l'adresse Ethernet source en **hexadécimal** : 00 06 29 9c 14 fe
- Quel est le type de la trame en **hexadécimal** : 08 00 La valeur hexadécimale 0800 indique que la trame contient un datagramme IP.
- Quelle est la version du protocole IP en **décimal** : 4 Le datagramme contenu correspond à la version 4 d'IP.
- Quelle est la longueur de l'entête IP en **décimal** : 5 5 * 4 = 20 octets, l'entête IP fait 20 octets, on se repère aussi avec l'adresse destination en fin d'entête IP
- Quelle est l'adresse IP source en **hexadécimal** : 0a 14 90 96 en faisant la traduction en décimal on a :
    - 0a = 0*16 + a soit 10.
    - 14 = 1*16 + 4 soit 20
    - 90 = 9*16 + 0 soit 144
    - 96 = 9*16 + 6 soit 150
    - on retrouve 10.20.144.150 qui est indiqué dans la fenêtre explicative et dans la colonne source de la capture Wireshark
- Quelle est l'adresse IP destination en **hexadécimal** : 0a 14 90 97 par le même calcul que ci-dessus, on retrouve 10.20.144.151 qui est aussi dans la colonne destination de la capture Wireshark

• Quel est le numéro du protocole indiqué par l'entête IP et transporté dans la charge utile du datagramme IP en **héxadécimal**, c'est TCP ou UDP : 06 cette valeur correspond à TCP. On trouve souvent 17 qui correspond à UDP. Pour avoir toute la liste, on peut accéder à https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml (vue le 21/07/2019).

• Quelle est la longueur totale du datagramme **en décimal** : 00 3c soit 3*16 + c = 60 octets

• Quel est le numéro de port source en **héxadécimal** : 8c 88 soit $8*16^3 + c*16^2 + 8*16 + 8 = 35976$, ce qui est bien ce qu'on retrouve ailleurs dans la trace de la trame 13

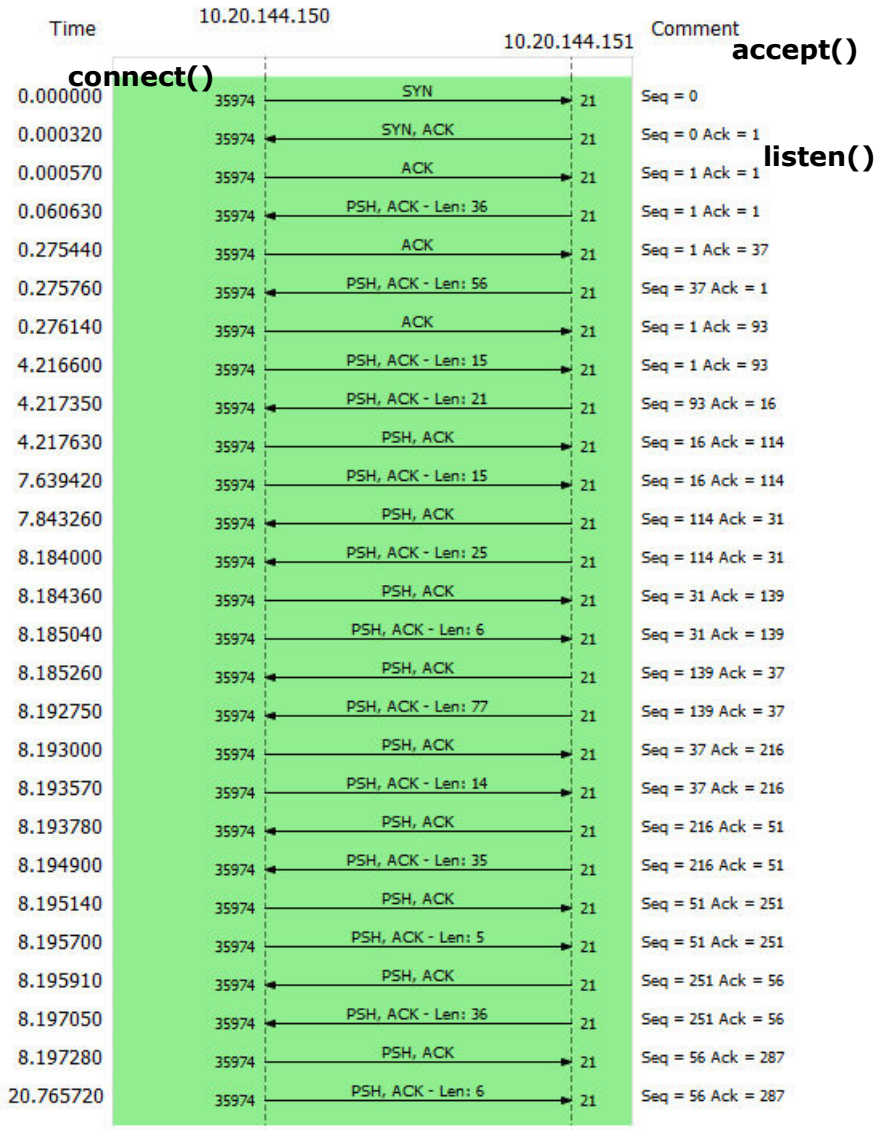
• Quel est le numéro de port destination en **héxadécimal** : 3e 8d soit $3*16^3 + e*16^2 + 8*16 + d = 16013$; ce qui correspond aussi

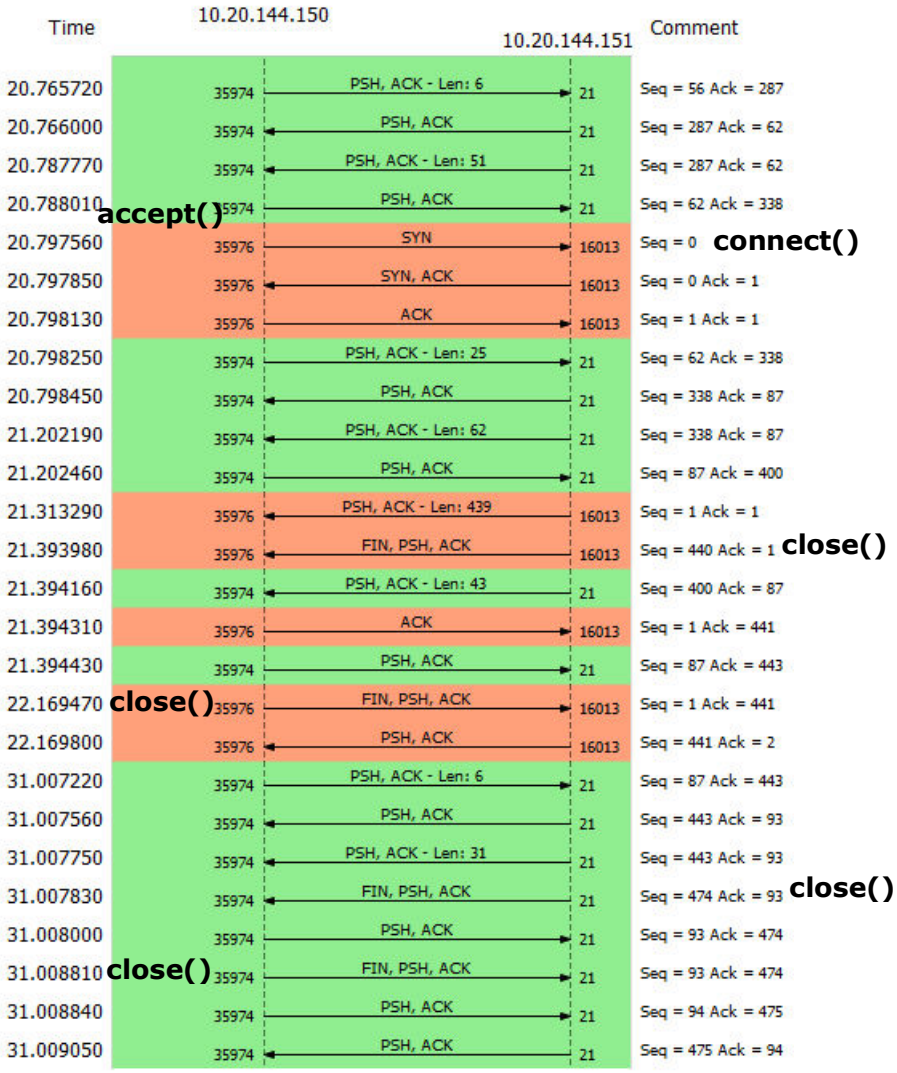
• Quel est la valeur du numéro de séquence absolu dans l'entête TCP en **hexadécimal** : 01 f9 8b d8 soit 33 131 480 en décimal

• Quel est le seul "flag" positionné dans l'entête TCP : 02 soit 0000 0010, ce qui correspond au bit indicateur SYN qui est le deuxième en partant de la droite ce champ.

Globalement, pour l'ensemble de ces informations, on peut les retrouver dans la partie explicitée de l'interface Wireshark. Ce qui donne un moyen de valider ce qu'on a trouver par ailleurs

Question 5 : On vous donne aussi le graphe synthétique des échanges ci-après. On a placé sur le graphe, autant de fois que nécessaire, les appels systèmes de l'API socket (en langage C pour une machine de type Unix). Y a-t-il des erreurs à propos de l'usage de ces primitives et au rôle client ou serveur, si oui, lesquelles, proposer une correction. Justifier brièvement votre réponse. **(2 points)**


Time
10.20.144.150
10.20.144.151
Comment
connect()
accept()
listen()
0.000000 35974 SYN 21 Seq = 0
0.000320 35974 SYN, ACK 21 Seq = 0 Ack = 1
0.000570 35974 ACK 21 Seq = 1 Ack = 1
0.060630 35974 PSH, ACK - Len: 36 21 Seq = 1 Ack = 1
0.275440 35974 ACK 21 Seq = 1 Ack = 37
0.275760 35974 PSH, ACK - Len: 56 21 Seq = 37 Ack = 1
0.276140 35974 ACK 21 Seq = 1 Ack = 93
4.216600 35974 PSH, ACK - Len: 15 21 Seq = 1 Ack = 93
4.217350 35974 PSH, ACK - Len: 21 21 Seq = 93 Ack = 16
4.217630 35974 PSH, ACK 21 Seq = 16 Ack = 114
7.639420 35974 PSH, ACK - Len: 15 21 Seq = 16 Ack = 114
7.843260 35974 PSH, ACK 21 Seq = 114 Ack = 31
8.184000 35974 PSH, ACK - Len: 25 21 Seq = 114 Ack = 31
8.184360 35974 PSH, ACK 21 Seq = 31 Ack = 139
8.185040 35974 PSH, ACK - Len: 6 21 Seq = 31 Ack = 139
8.185260 35974 PSH, ACK 21 Seq = 139 Ack = 37
8.192750 35974 PSH, ACK - Len: 77 21 Seq = 139 Ack = 37
8.193000 35974 PSH, ACK 21 Seq = 37 Ack = 216
8.193570 35974 PSH, ACK - Len: 14 21 Seq = 37 Ack = 216
8.193780 35974 PSH, ACK 21 Seq = 216 Ack = 51
8.194900 35974 PSH, ACK - Len: 35 21 Seq = 216 Ack = 51
8.195140 35974 PSH, ACK 21 Seq = 51 Ack = 251
8.195700 35974 PSH, ACK - Len: 5 21 Seq = 51 Ack = 251
8.195910 35974 PSH, ACK 21 Seq = 251 Ack = 56
8.197050 35974 PSH, ACK - Len: 36 21 Seq = 251 Ack = 56
8.197280 35974 PSH, ACK 21 Seq = 56 Ack = 287
20.765720 35974 PSH, ACK - Len: 6 21 Seq = 56 Ack = 287


Time
10.20.144.150
10.20.144.151
Comment
20.765720 35974 PSH, ACK - Len: 6 21 Seq = 56 Ack = 287
20.766000 35974 PSH, ACK 21 Seq = 287 Ack = 62
20.787770 35974 PSH, ACK - Len: 51 21 Seq = 287 Ack = 62
20.788010 35974 PSH, ACK 21 Seq = 62 Ack = 338
accept()
20.797560 35976 SYN 16013 Seq = 0
connect()
20.797850 35976 SYN, ACK 16013 Seq = 0 Ack = 1
20.798130 35976 ACK 16013 Seq = 1 Ack = 1
20.798250 35974 PSH, ACK - Len: 25 21 Seq = 62 Ack = 338
20.798450 35974 PSH, ACK 21 Seq = 338 Ack = 87
21.202190 35974 PSH, ACK - Len: 62 21 Seq = 338 Ack = 87
21.202460 35974 PSH, ACK 21 Seq = 87 Ack = 400
21.313290 35976 PSH, ACK - Len: 439 16013 Seq = 1 Ack = 1
21.393980 35976 FIN, PSH, ACK 16013 Seq = 440 Ack = 1
close()
21.394160 35974 PSH, ACK - Len: 43 21 Seq = 400 Ack = 87
21.394310 35976 ACK 16013 Seq = 1 Ack = 441
21.394430 35974 PSH, ACK 21 Seq = 87 Ack = 443
22.169470 35976 FIN, PSH, ACK 16013 Seq = 1 Ack = 441
close()
22.169800 35976 PSH, ACK 16013 Seq = 441 Ack = 2
31.007220 35974 PSH, ACK - Len: 6 21 Seq = 87 Ack = 443
31.007560 35974 PSH, ACK 21 Seq = 443 Ack = 93
31.007750 35974 PSH, ACK - Len: 31 21 Seq = 443 Ack = 93
31.007830 35974 FIN, PSH, ACK 21 Seq = 474 Ack = 93
close()
31.008000 35974 PSH, ACK 21 Seq = 93 Ack = 474
31.008810 35974 FIN, PSH, ACK 21 Seq = 93 Ack = 474
close()
31.008840 35974 PSH, ACK 21 Seq = 94 Ack = 475
31.009050 35974 PSH, ACK 21 Seq = 475 Ack = 94

Correction :

listen()

accept()

connect()

Time	10.20.144.150		10.20.144.151	Comment
0.000000	35974	SYN →	21	Seq = 0
0.000320	35974	← SYN, ACK	21	Seq = 0 Ack = 1
0.000570	35974	ACK →	21	Seq = 1 Ack = 1
0.060630	35974	← PSH, ACK - Len: 36	21	Seq = 1 Ack = 1
0.275440	35974	ACK →	21	Seq = 1 Ack = 37
0.275760	35974	← PSH, ACK - Len: 56	21	Seq = 37 Ack = 1
0.276140	35974	ACK →	21	Seq = 1 Ack = 93
4.216600	35974	PSH, ACK - Len: 15 →	21	Seq = 1 Ack = 93
4.217350	35974	← PSH, ACK - Len: 21	21	Seq = 93 Ack = 16
4.217630	35974	PSH, ACK →	21	Seq = 16 Ack = 114
7.639420	35974	PSH, ACK - Len: 15 →	21	Seq = 16 Ack = 114
7.843260	35974	← PSH, ACK	21	Seq = 114 Ack = 31
8.184000	35974	← PSH, ACK - Len: 25	21	Seq = 114 Ack = 31
8.184360	35974	PSH, ACK →	21	Seq = 31 Ack = 139
8.185040	35974	PSH, ACK - Len: 6 →	21	Seq = 31 Ack = 139
8.185260	35974	← PSH, ACK	21	Seq = 139 Ack = 37
8.192750	35974	← PSH, ACK - Len: 77	21	Seq = 139 Ack = 37
8.193000	35974	PSH, ACK →	21	Seq = 37 Ack = 216
8.193570	35974	PSH, ACK - Len: 14 →	21	Seq = 37 Ack = 216
8.193780	35974	← PSH, ACK	21	Seq = 216 Ack = 51
8.194900	35974	← PSH, ACK - Len: 35	21	Seq = 216 Ack = 51
8.195140	35974	PSH, ACK →	21	Seq = 51 Ack = 251
8.195700	35974	PSH, ACK - Len: 5 →	21	Seq = 51 Ack = 251
8.195910	35974	← PSH, ACK	21	Seq = 251 Ack = 56
8.197050	35974	← PSH, ACK - Len: 36	21	Seq = 251 Ack = 56
8.197280	35974	PSH, ACK →	21	Seq = 56 Ack = 287
20.765720	35974	PSH, ACK - Len: 6 →	21	Seq = 56 Ack = 287

Time	10.20.144.150		10.20.144.151	Comment
20.765720	35974	PSH, ACK - Len: 6 →	21	Seq = 56 Ack = 287
20.766000	35974	← PSH, ACK	21	Seq = 287 Ack = 62
20.787770	35974	← PSH, ACK - Len: 51	21	Seq = 287 Ack = 62
20.788010	35974	PSH, ACK →	21	Seq = 62 Ack = 338
20.797560	35976	SYN →	16013	Seq = 0
20.797850	35976	← SYN, ACK	16013	Seq = 0 Ack = 1
20.798130	35976	ACK →	16013	Seq = 1 Ack = 1
20.798250	35974	PSH, ACK - Len: 25 →	21	Seq = 62 Ack = 338
20.798450	35974	← PSH, ACK	21	Seq = 338 Ack = 87
21.202190	35974	← PSH, ACK - Len: 62	21	Seq = 338 Ack = 87
21.202460	35974	PSH, ACK →	21	Seq = 87 Ack = 400
21.313290	35976	← PSH, ACK - Len: 439	16013	Seq = 1 Ack = 1
21.393980	35976	← FIN, PSH, ACK	16013	Seq = 440 Ack = 1
21.394160	35974	← PSH, ACK - Len: 43	21	Seq = 400 Ack = 87
21.394310	35976	ACK →	16013	Seq = 1 Ack = 441
21.394430	35974	PSH, ACK →	21	Seq = 87 Ack = 443
22.169470	35976	FIN, PSH, ACK →	16013	Seq = 1 Ack = 441
22.169800	35976	← PSH, ACK	16013	Seq = 441 Ack = 2
31.007220	35974	PSH, ACK - Len: 6 →	21	Seq = 87 Ack = 443
31.007560	35974	← PSH, ACK	21	Seq = 443 Ack = 93
31.007750	35974	← PSH, ACK - Len: 31	21	Seq = 443 Ack = 93
31.007830	35974	← FIN, PSH, ACK	21	Seq = 474 Ack = 93
31.008000	35974	PSH, ACK →	21	Seq = 93 Ack = 474
31.008810	35974	FIN, PSH, ACK →	21	Seq = 93 Ack = 474
31.008840	35974	PSH, ACK →	21	Seq = 94 Ack = 475
31.009050	35974	← PSH, ACK	21	Seq = 475 Ack = 94

accept()

connect()

close()

close()

close()

Plusieurs erreurs :

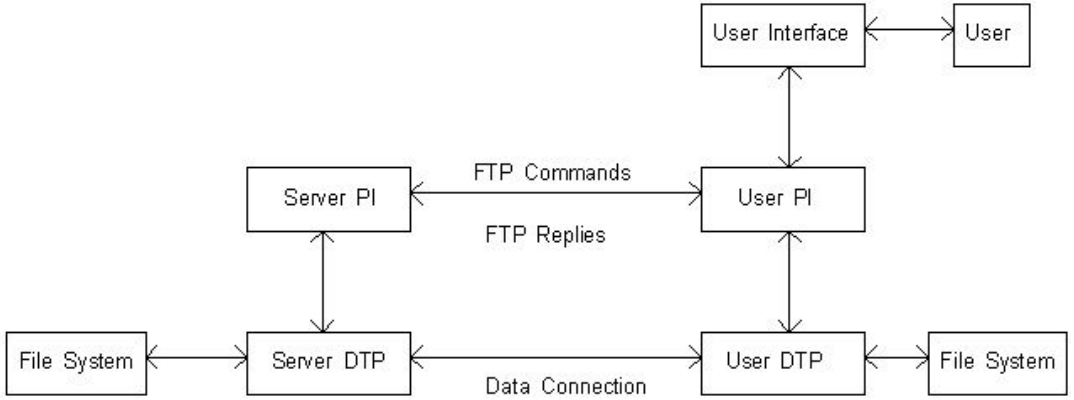
- Côté serveur :
  - listen () ne peut pas être avant l'accept() car l'appel listen() définit la taille de la file d'attente des demandes d'ouverture de connexion.
  - L'accept() est fait bien en amont, pour préparer l'ouverture de connexion et être prêt à répondre au three way handshake. Comme c'est une préparation, on dit que c'est le côté ouverture de connexion passive.
  - Il ne peut y avoir un connect() côté serveur, enfin, c'est possible, mais ce n'est pas le mode d'utilisation le plus courant.
  - Les close() sont à leur place, il ont été produit afin que la connexion se ferme dans TCP par un FIN
  - On remarque que pour les 2 connexions, les fermetures sont à l'initiative du serveur
- Côté client :
  - Il y a un accept() là où il devrait y avoir un connect()
  - Il n'y a pas d'autre erreur

Question 6 : Le texte ci-dessous représente des échanges entre le client et le serveur. Ce sont des échanges protocolaires de la couche 7 du modèle OSI. Commenter ce qui se passe entre le client et le serveur. **(1 point)**

```
220-QTCP at fran.csg.stercomm.com.
220 Connection will close if idle more than 5 minutes.
USER cdts3500
331 Enter password.
PASS cdts3500
230 CDTS3500 logged on.
SYST
215 OS/400 is the remote operating system. The TCP/IP version is "V5R2M0".
SITE NAMEFMT
250 Now using naming format "0".
PWD
257 "CDTS3500" is current library.
PASV
227 Entering Passive Mode (10,20,144,151,62,141).
RETR qgpl/apkeyf.apkeyf
150 Retrieving member APKEYF in file APKEYF in library QGPL.
250 File transfer completed successfully.
QUIT
221 QUIT subcommand received.
```

Correction :
Les échanges observés ressemblent à une interaction client-serveur pour l'échange d'un fichier "File transfer completed successsfully". On voit un login/password, puis un retrait de fichier, puis une déconnexion.

Le protocole FTP et son implantation peuvent se résumer sur la figure suivante :



source : http://www.rhyshaden.com/ftp.htm (consulté le 21/07/2019)

PI signifie Protocol Interpreter; DTP signifie Data Transfer Process

On reprend la suite d'échanges en la commentant.

1. **220-QTCP at fran.csg.stercomm.com.**
   Connexion TCP réussie avec la machine fran.csg.stercomm.com, envoi par le serveur au client, ceci correspond à la ligne/trame 4 de la trace Wireshark
   220 est un code de retour, en l'occurrence ça se passe bien
2. **220 Connection will close if idle more than 5 minutes.**
   Serveur vers le client (ligne 6 de la trace), si la connexion est inactive plus de 5mn elle sera fermée
3. **USER cdts3500**
   le client sollicite le serveur (ligne 8), il envoie le nom de login, ici cdts3500
4. **331 Enter password.**
   Le serveur répond au client (ligne 9) et exige le mot de passe
5. **PASS cdts3500**
   Le client envoie le mot de passe (ligne 11) : cdts3500 qui est le nom de login (c'est une très mauvaise pratique du point

de vue de la sécurité)

```
6.  230 CDTS3500 logged on.
```

le serveur indique au client qu'il a ouvert sa session avec succès (ligne 13)

```
7.  SYST
```

Le client demande au serveur d'afficher le type du système (ligne 15)

```
8.  215 OS/400 is the remote operating system. The TCP/IP version is "V5R2M0".
```

le serveur envoie l'information demandée (ligne 17 de la trace)

```
9.  SITE NAMEFMT
```

Le client demande le format du nom de fichier à utiliser (ligne 19 de la trace) NAMEFMT est une sous-commande IBM d'un client FTP qui sélectionne quel est le format du nom de fichier à utiliser sur le système local et sur celui du serveur distant. Le système de nommage dans l'aborescence des fichiers doit être propre à IBM.

```
10. 250 Now using naming format "0".
```

Réponse du serveur (ligne 21). La réponse étant "0", le format utilisé est le suivant [libname/]filename[.mbrname], c'est semble-t-il un format lié à des bases de données.

```
11. PWD
```

Le client demande le répertoire courant sur le serveur (ligne 23).

```
12. 257 "CDTS3500" is current library.
```

Le serveur indique que CDTS3500 est le répertoire courant (ligne 25).

```
13. PASV
```

Le client indique un passage en mode passif. Il attend que le serveur lui fournisse une adresse et un port pour accéder à un fichier (ligne 27).

Pour le mode actif, il spécifierait un numéro de port avec la commande PORT pour indiquer sur quel port il se mettrait en écoute pour recevoir des données.

```
14. 227 Entering Passive Mode (10,20,144,151,62,141).
```

Le serveur bascule en mode passif. Il spécifie l'adresse et le numéro de port de la connexion qui devront être utilisés par le client (ligne 29).

Les 4 premiers champs donnent l'adresse IP, ici 10.20.144.151, on reconnaît l'adresse IP du serveur. On comprend aussi que dans la suite de la trace, ligne 31, c'est le client qui fait une ouverture de connexion active.

Les deux champs suivants servent au calcul du numéro de port de la façon suivante : 62 * 256 + 141, soit 16013 qu'on a bien dans la trace Wireshark comme port serveur de la 2ème connexion TCP.

La première connexion TCP a servi au mode commande, tandis que la deuxième sert au mode transfert de données.

```
15. RETR qgpl/apkeyf.apkeyf
```

Le client demande à récupérer le fichier qgpl/apkeyf.apkeyf (ligne 34).

16. **150 Retrieving member APKEYF in file APKEYF in library QGPL.**

Le serveur transfère le ficher demandé vers le client (ligne 36). Le transfert effectif est fait sur la connexion de transfert de données (la 2ème ouverte) (ligne 38) avec la commande "509 FTP Data…" qui d'ailleurs indique que 439 octets ont été transférés vers le client. La connexion de données est fermée immédiatement après, à l'initiative du serveur (ligne/trame 39).

17. **250 File transfer completed successfully.**

Le serveur indique que le transfert s'est effectué avec succès (ligne 40). On est sur la connexion d'échange de commandes qui est établie entre les extrémités (10.20.144.150, 39674) et (10.20.144.151, 21) depuis le début de la trace.

18. **QUIT**

Le client demande à fermer la session (ligne 45)

19. **221 QUIT subcommand received.**

Fermeture de session par le serveur (ligne/trame 47)

Pour une liste complète des commandes, consulter le lien https://fr.wikipedia.org/wiki/Liste_des_commandes_ftp et pour la liste des codes https://fr.wikipedia.org/wiki/Liste_des_codes_des_r%C3%A9ponses_d%27un_serveur_FTP (accédés le 21/07/2019). Les codes en 2xx correspondent à des succès.

Question 7 : Combien d'octets échangés pour chaque sens dans la connexion la plus interne qui démarre au temps t=20.797560ms, qui se termine au temps t=22.169800ms dans le graphe. Cet échange est, d'ailleurs, marqué en couleur rosée dans le graphe des échanges. Expliquer votre raisonnement. **(2 points)**

Correction :

Les connexions TCP sont full duplex, donc il y a échange des informations dans les deux sens : client vers serveur, et serveur vers client. On ne considère que la connexion la plus interne, elle commence à la trame 31, et elle se termine à la trame 43 acquittée en trame 44. Les ports impliqués sur cette connexion sont 35976 et 16013.

Données échangées dans le sens client (35976) vers serveur (16013) :

Le serveur envoie un segment avec no seq 441, et un no ack 2 en ligne 44. Cela veut dire qu'il a reçu des segments jusqu'à l'octet numéro 1 car il attend l'octet numéro 2. Comme le SYN compte comme un octet (qui porte le numéro 0 en numérotation relative) et que le FIN aussi (qui porte le numéro 1). Il n'y a pas de données échangées du client vers le serveur.

Données échangées dans le sens serveur (16013) vers client (35976) :
Dans le sens serveur vers client, on a ligne 44 un segment avec no seq 1 et no ack 441. Le client a donc reçu les octets de numéros 0 à 440. Le SYN et le FIN comptent chacun pour un octet. Les octets de données reçus vont par conséquent de 1 à 439. Le client a donc reçu 439 octets.

On vous donne l'extrait du résultat de la commande `ipconfig` sur l'équipement d'adresse IP 10.20.144.150 :

```
Carte Ethernet Connexion au réseau local filaire eth0 :
   Suffixe DNS propre à la connexion. . . : mbi.moc
   Adresse IPv4. . . . . . . . . . . . . .: 10.20.144.150
   Masque de sous-réseau. . . . . . . . . : 255.255.0.0
   Passerelle par défaut. . . . . . . . . : 10.20.144.202

Carte réseau sans fil Connexion réseau sans fil wif0 :
   Suffixe DNS propre à la connexion. . . : mbi.moc
   Adresse IPv4. . . . . . . . . . . . . .: 10.30.144.050
   Masque de sous-réseau. . . . . . . . . : 255.255.0.0
   Passerelle par défaut. . . . . . . . . : 10.30.144.252
```

Question 8 : Paramètres de configuration de l'équipement **(1 point)**

- Donner la notation compacte pour le masque en /n. **(0,25 point)**

Correction : 255.255.0.0 correspond à 16 bits à 1 en partant de la gauche. La notation compacte est donc /16

- Donner l'adresse de réseau IP associée correspondant à chacune des interfaces. **(0,25 point)**

Correction :
Pour l'interface Ethernet, l'addresse IP 10.20.144.150 qui correspond au réseau 10.20.0.0/16
Pour l'interface Wifi, l'addresse IP 10.30.144.050 qui correspond au réseau 10.30.0.0/16

- Donner l'adresse IP du routeur associé pour chacun des réseaux IP auxquels ces interfaces appartiennent. **(0,25 point)**

Correction :
L'adresse IP du routeur, associé au réseau 10.20.0.0/16 auquel la carte Ethernet est raccordée, est 10.20.144.202
L'adresse IP du routeur, associé au réseau 10.30.0.0/16 auquel la carte Wifi est raccordée, est 10.20.144.252

- Donner l'adresse de diffusion associée à chacun des réseaux IP apparaissant dans les résultats ci-dessus. **(0,25 point)**

Correction :
Pour l'interface Ethernet, l'addresse IP de broadcast est 10.20.255.255, elle correspond au réseau 10.20.0.0/16
Pour l'interface Wifi, l'addresse IP de broadcast est 10.30.255.255, elle correspond au réseau 10.30.0.0/16

Question 9 : Quand l'équipement met en fonctionnement uniquement sa carte Ethernet on a la table de routage locale suivante active:

	Réseau/mask	Next hop	métrique	accessibilité	interface
L1	0.0.0.0/0	10.20.144.202	10	distant	eth0
L2	127.0.0.0/8	0.0.0.0	0	direct	lo0
L3	10.20.0.0/16	0.0.0.0	0	direct	eth0

Quelle ligne de la table de routage emprunte le datagramme d'adresse IP destination 10.30.144.10 pour sortir du routeur et atteindre cette destination ? Expliquez brièvement votre réponse. **(1 point)**
Correction :
L1 : 10.30.144.10/0 donne 0.0.0.0, L1 est donc candidate
L2 : 10.30.144.10/8 donne 10.0.0.0, ça ne correspond pas du tout à 127.0.0.0/8
L3 : 10.30.144.10/16 donne 10.20.0.0, ça ne correspond pas du tout à l'adresse réseau IP associée à L3
L1 étant la seule ligne candidate restant, c'est L1 qui fera sortir le datagramme d'adresse destination 10.30.144.10.

Question 10 : En reprenant la trame 31, que se passe-t-il si le datagramme doit être fragmenté lors de la traversée du routeur ? **(1 point)**
Correction :
Dans la trame 1, dans l'entête IP, l'indicateur "don't fragment" est positionné. Par conséquent, si un routeur doit fragmenter le datagramme pour qu'il poursuive son chemin, il ne le fera pas, et indiquera une erreur à l'émetteur du datagramme.

Exercice 2 : Question de cours (3 points)

1. Expliquez sommairement la différence entre SEM et TSCM. Pour chacun, fournir un exemple concret de problème repéré par ces actions de surveillance.

Correction :
SEM = Security Event Monitoring, surveillance de sécurité.
Exemple d'événement surveillé/repéré : création/suppression d'un compte utilisateur, logins ou échecs d'authentification, modification de fichiers de configuration critiques du système/d'une application, tcpdump et sniffing réseau, chargement de pilotes de périphériques/device drivers, etc. (via l'audit)

TSCM = Technical State Compliance Monitoring: vérification que les systèmes restent dans la configuration dans laquelle ils sont supposés être sur le long terme.
Par exemple : IPv6 désactivé, pas de compilateur/d'outil de développement installé en Production, ou de bibliothèques X11 sous Unix (par exemple).

2. Pourquoi a-t-on besoin de faire certifier les clés publiques en cryptographie à clé publique ?

Correction :
En crytographie à clé publique, on n'a pas besoin d'un canal de communication sécurité pour échanger les clés publiques entre participants, car elles sont publiques. Mais comme rien ne ressemble plus à une clé publique qu'une autre, si un intercepteur actif remplace la clé publique de Bob par la sienne, Alice chiffrera son message à l'attention de l'intercepteur au lieu de le chiffrer à l'attention de Bob.
On a donc besoin d'un tiers pour certifier l'appartenance des clés publiques pour s'en prémunir.

3. Pourquoi est-ce inutile en cryptographie symétrique ? Dans ce cas, expliquez comment on s'assure d'utiliser la bonne clé.

Correction :
En cryptographie symétrique, Alice et Bob doivent se rencontrer et échanger une clé secrète (pré-partagée) long-terme avant de pouvoir commencer à chiffrer.
La cryptographie symétrique n'offre pas d'alternative : c'est le rôle de la valise diplomatique. Il appartient aux participants (Alice, Bob) de s'assurer eux-mêmes de l'identité de l'autre participant.

Remarque : Seul Kerberos a une solution à ce problème, mais Kerberos n'est pas traité en UTC505. Dans Kerberos, c'est le KDC (Key Distribution Center), Gardien des clés, qui assigne les clés de session à tous les utilisateurs et services, donc qui concentre la sécurité du système.

Exercice 3 : Sécurité inconditionnelle (2 points)

La sécurité inconditionnelle du masque à usage unique a été expliquée en cours via un exemple où l'on chiffre le message « SECRET » avec la clé « XOSAUF » pour obtenir « PSURYY ». À partir du moment où l'algorithme est respecté, et donc où la clé de chiffrement a été irrémédiablement détruite immédiatement après utilisation, le cryptogramme « PSURYY » est indéchiffrable, quel que soit le temps et la puissance informatique qu'on y consacre.

Expliquez pourquoi ce système est inconditionnellement sûr, en français compréhensible (pas d'équation mathématique) et 6 lignes au maximum.

Correction :
L'algorithme est inconditionnellement sûr car tous les déchiffrements sont possibles et équiprobables statistiquement, donc on ne pourra jamais savoir quel est le bon déchiffrement. (Pour chaque déchiffrement possible, il existe une unique clé permettant de déchiffrer le cryptogramme en ce déchiffrement).

Comme réponse à cette question tout argument qui est également valable pour tout algorithme symétrique (qui eux ne sont pas inconditionnellement sûrs) n'était pas acceptable, comme par exemple :

- Le système est sûr car les clés sont choisies aléatoirement
- Ou car il y a un grand nombre de clés possibles
- Etc.