

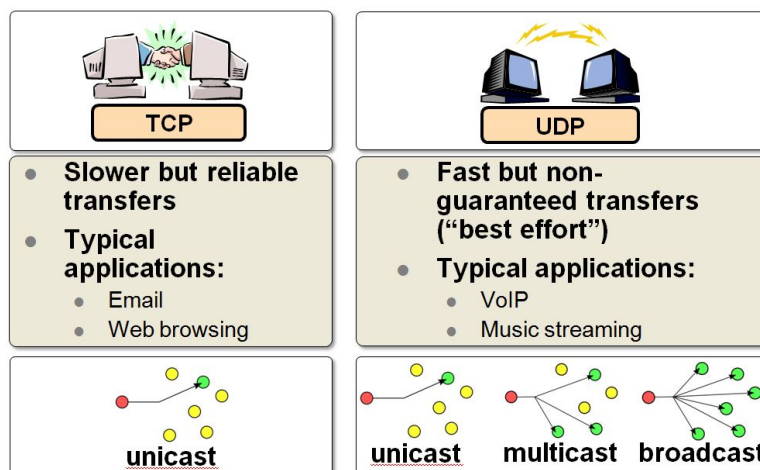


.....TCP-UDP 15.01.2024

C'est en segmentant que ça fait un paquet entier☺

On se connecte ou pas ?

Auteur : Pascal Fougeray



source : <https://geekflare.com/fr/tcp-vs-udp/>

1 Introduction

La **couche 4** du modèle OSI, la couche **transport** où l'unité des données est le **segment**.

Pourquoi segment ?

Tout simplement parce qu'il est **impossible parfois** de recevoir ou d'envoyer un **paquet IP** en une seule **trame Ethernet** à cause du MTU.

Donc on fait ce que l'on appelle de la **segmentation**

Dans le cours sur les ports et les services nous avons vu (ou nous verrons) les sockets.

TCP c'est les sockets de type **STREAM** ou flux de données on dit que c'est un mode **connecté**

UDP c'est les sockets de type **DATAGRAM** ou paquets de données on dit que c'est un mode **déconnecté**

Il existe aussi les sockets de type **RAW** pour **ICMP** et **ARP**.

ICMP et ARP ne sont ni TCP ni UDP

2 Question et Réponse

Comme vous allez me la poser souvent et que souvent je vais devoir y répondre, je pose la question et j'y réponds tout de suite.

Quand utiliser UDP et quand utiliser TCP ?

Réponse :

— On utilise UDP quand on fait une demande et qu'on attend qu'une seule réponse.

Par exemple les protocoles



DHCP où l'on demande une @réseau IP, un masque (mask), et éventuellement l'adresse d'un serveur DNS ce qui généralement est de taille inférieure à un segment

La capture wireshark ci-dessous montre un échange qui a commencé au bout de 9,45s et qui s'est terminé au bout de 10,48 soit 1s d'échanges.

No.	Time	Source	Destination	Protocol	Length	Info
7	9.457673	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xdb0e344c
10	10.458712	172.31.1.1	172.31.1.12	DHCP	342	DHCP Offer - Transaction ID 0xdb0e344c
11	10.477603	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0xdb0e344c
13	10.486062	172.31.1.1	172.31.1.12	DHCP	342	DHCP ACK - Transaction ID 0xdb0e344c

<p>Frame 10: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface -, id 0</p> <p>Ethernet II, Src: 3e:6b:24:a4:c7:6f (3e:6b:24:a4:c7:6f), Dst: 36:d8:a2:17:24:8f (36:d8:a2:17:24:8f)</p> <p>Internet Protocol Version 4, Src: 172.31.1.1, Dst: 172.31.1.12</p> <p>User Datagram Protocol, Src Port: 67, Dst Port: 68</p> <p>Dynamic Host Configuration Protocol (Offer)</p> <p>Message type: Boot Reply (2)</p> <p>Hardware type: Ethernet (0x01)</p> <p>Hardware address length: 6</p> <p>Hops: 0</p> <p>Transaction ID: 0xdb0e344c</p> <p>Seconds elapsed: 0</p> <p>Bootp flags: 0x0000 (Unicast)</p> <p>Client IP address: 0.0.0.0</p> <p>Your (client) IP address: 172.31.1.12</p> <p>Next server IP address: 172.31.1.1</p> <p>Relay agent IP address: 0.0.0.0</p> <p>Client MAC address: 36:d8:a2:17:24:8f (36:d8:a2:17:24:8f)</p> <p>Client hardware address padding: 000000000000000000000000</p>

DNS où l'on demande simplement la correspondance entre un nom de serveur tel **www.unicaen.fr** et qu'il récupère l'IP **193.55.120.26** correspondant comme on peut le voir ci-dessous.

```
root@PAF:~# nslookup www.unicaen.fr
Server:      127.0.0.53
Address:     127.0.0.53#53
```

```
Non-authoritative answer:
www.unicaen.fr canonical name = rp5.unicaen.fr.
Name:   rp5.unicaen.fr
Address: 193.55.120.26
```

Ce n'est juste qu'une requête avec une simple réponse.
Nous étudierons DNS et DHCP dans un cours ultérieur

- On utilise TCP quand il va y avoir plusieurs échanges par exemple SMTP (mail) , SSH, HTTP etc...
- Voici un exemple que nous ferons sur le TP protocole.

On saisit un ensemble de commandes telles
telnet localhost 25 -b 192.168.56.1,

1. **HELO Bonjour Prof,**
2. **MAIL FROM :<Moi@NewYork.com> ,**
3. **RCPT TO :<pascal@PAF.lan> ,**
4. **DATA**
5. **Bonjour les L3 et Bonne Année 2023;)**
6. **.**
7. **QUIT**

Chaque commande (ici 7) engendre un échange, **donc un dialogue long entre le client et le serveur.**

```
pascal@PAF:~# telnet localhost 25 -b 192.168.56.1
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 PAF.lan ESMTP Postfix (Ubuntu)
HELO Bonjour Prof
250 PAF.lan
MAIL FROM:<Moi@NewYork.com>
250 2.1.0 Ok
```



```

RCPT TO:<pascal@PAF.lan>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Bonjour les L3 et Bonne Année 2023 ;)
.
250 2.0.0 Ok: queued as DCF3DE20CF
QUIT
221 2.0.0 Bye
Connection closed by foreign host.

```

La capture wireshark ci-dessous montre un échange qui a duré plus de 2mn (131s).
2mn pendant lesquelles le client et le serveur étaient connectés

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	192.168.56.1	127.0.0.1	TCP	74	53409 → 25 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=2193776507 TSecr=0 WS=128
2	0.00004104	127.0.0.1	192.168.56.1	TCP	74	25 → 53409 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=4155130325 TSecr=
3	0.000076079	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2193776507 TSecr=4155130325
6	0.059741452	127.0.0.1	192.168.56.1	SMTP	102	S: 220 PAF.lan ESMTP Postfix (Ubuntu)
7	0.059754836	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=1 Ack=37 Win=65536 Len=0 TSval=2193776507 TSecr=4155130385
8	24.324976847	192.168.56.1	127.0.0.1	SMTP	85	C: HELO Bonjour Prof
9	24.325020545	127.0.0.1	192.168.56.1	TCP	66	25 → 53409 [ACK] Seq=37 Ack=20 Win=65536 Len=0 TSval=4155154650 TSecr=2193800832
10	24.325191876	127.0.0.1	192.168.56.1	SMTP	79	S: 250 PAF.lan
11	24.325219833	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=20 Ack=50 Win=65536 Len=0 TSval=2193800832 TSecr=4155154650
12	62.662757505	192.168.56.1	127.0.0.1	SMTP	95	C: MAIL FROM:<Moi@NewYork.com>
13	62.673326475	127.0.0.1	192.168.56.1	SMTP	89	S: 250 2.1.0 Ok
14	62.673339170	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=49 Ack=64 Win=65536 Len=0 TSval=2193839180 TSecr=4155192998
15	104.68403171	192.168.56.1	127.0.0.1	SMTP	92	C: RCPT TO:<pascal@PAF.lan>
16	104.69707144	127.0.0.1	192.168.56.1	SMTP	89	S: 250 2.1.5 Ok
17	104.697086352	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=75 Ack=78 Win=65536 Len=0 TSval=2193881204 TSecr=4155235022
18	112.350923073	192.168.56.1	127.0.0.1	SMTP	72	C: DATA
19	112.351233802	127.0.0.1	192.168.56.1	SMTP	103	S: 354 End data with <CR><LF>.<CR><LF>
20	112.35126455	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=81 Ack=115 Win=65536 Len=0 TSval=2193888858 TSecr=4155242676
21	126.5660017	192.168.56.1	127.0.0.1	SMTP	106	C: DATA fragment, 40 bytes
22	126.6020384	127.0.0.1	192.168.56.1	TCP	66	25 → 53409 [ACK] Seq=115 Ack=121 Win=65536 Len=0 TSval=4155256927 TSecr=2193903067
23	128.3330876	192.168.56.1	127.0.0.1	SMTP	69	Bonjour les L3 et Bonne Année 2023 ;)
24	128.3331218	127.0.0.1	192.168.56.1	TCP	66	25 → 53409 [ACK] Seq=115 Ack=124 Win=65536 Len=0 TSval=4155258658 TSecr=2193904840
25	128.3413145	127.0.0.1	192.168.56.1	SMTP	103	S: 250 2.0.0 Ok: queued as DCF3DE20CF
26	128.3413494	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=124 Ack=152 Win=65536 Len=0 TSval=2193904848 TSecr=4155258666
27	131.5809863	192.168.56.1	127.0.0.1	SMTP	72	C: QUIT
28	131.5815543	127.0.0.1	192.168.56.1	SMTP	81	S: 221 2.0.0 Bye
29	131.5815850	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [ACK] Seq=130 Ack=167 Win=65536 Len=0 TSval=2193908089 TSecr=4155261907
30	131.5816240	127.0.0.1	192.168.56.1	TCP	66	25 → 53409 [FIN, ACK] Seq=167 Ack=130 Win=65536 Len=0 TSval=4155261907 TSecr=2193908089
31	131.5818787	192.168.56.1	127.0.0.1	TCP	66	53409 → 25 [FIN, ACK] Seq=130 Ack=168 Win=65536 Len=0 TSval=2193908089 TSecr=4155261907
32	131.5819342	127.0.0.1	192.168.56.1	TCP	66	25 → 53409 [ACK] Seq=168 Ack=131 Win=65536 Len=0 TSval=4155261907 TSecr=2193908089

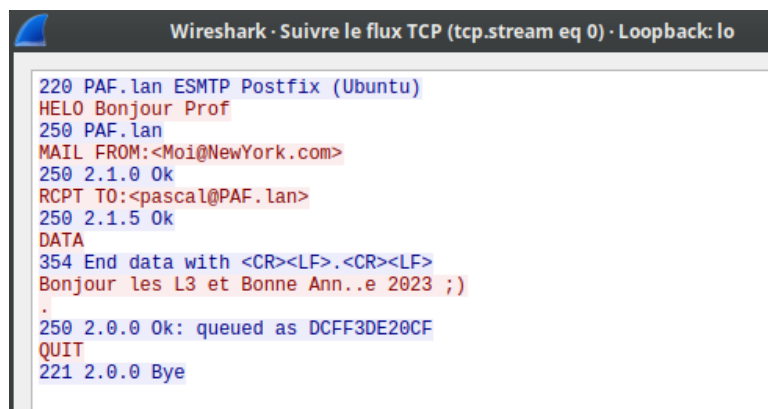
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
 Internet Protocol Version 4, Src: 192.168.56.1, Dst: 127.0.0.1
 Transmission Control Protocol, Src Port: 53409, Dst Port: 25, Seq: 0, Len: 0

Si on fait une **analyse de flux** avec wireshark on obtient ceci

Temps	192.168.56.1	127.0.0.1	127.0.0.53	Commentaire
0.000000000	53409 → 25 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SA	25		TCP: 53409 → 25 [SYN] Seq=0 Win=65495 Len=0 MSS=65495...
0.000041041	25 → 53409 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MS	25		TCP: 25 → 53409 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 ...
0.000076079	53409 → 25 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2	25		TCP: 53409 → 25 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=...
0.043256560	34884 Standard query 0x582c PTR 1.56.168.192.in-addr.arpa OPT	53		DNS: Standard query 0x582c PTR 1.56.168.192.in-addr.arpa ...
0.050025295	34884 Standard query response 0x582c PTR 1.56.168.192.in-addr	53		DNS: Standard query response 0x582c PTR 1.56.168.192.in-...
0.059741452	S: 220 PAF.lan ESMTP Postfix (Ubuntu)	25		SMTP: S: 220 PAF.lan ESMTP Postfix (Ubuntu)
0.059754836	53409 → 25 [ACK] Seq=1 Ack=37 Win=65536 Len=0 TSval=	25		TCP: 53409 → 25 [ACK] Seq=1 Ack=37 Win=65536 Len=0 TSv...
24.324976847	C: HELO Bonjour Prof	25		SMTP: C: HELO Bonjour Prof
24.325020545	25 → 53409 [ACK] Seq=37 Ack=20 Win=65536 Len=0 TSval=	25		TCP: 25 → 53409 [ACK] Seq=37 Ack=20 Win=65536 Len=0 TS...
24.325191876	S: 250 PAF.lan	25		SMTP: S: 250 PAF.lan
24.325219833	53409 → 25 [ACK] Seq=20 Ack=50 Win=65536 Len=0 TSval=	25		TCP: 53409 → 25 [ACK] Seq=20 Ack=50 Win=65536 Len=0 TS...
62.662757505	C: MAIL FROM:<Moi@NewYork.com>	25		SMTP: C: MAIL FROM:<Moi@NewYork.com>
62.673326475	S: 250 2.1.0 Ok	25		SMTP: S: 250 2.1.0 Ok
62.673339170	53409 → 25 [ACK] Seq=49 Ack=64 Win=65536 Len=0 TSval=	25		TCP: 53409 → 25 [ACK] Seq=49 Ack=64 Win=65536 Len=0 TS...
104.684031731	C: RCPT TO:<pascal@PAF.lan>	25		SMTP: C: RCPT TO:<pascal@PAF.lan>
104.697071444	S: 250 2.1.5 Ok	25		SMTP: S: 250 2.1.5 Ok
104.697086352	53409 → 25 [ACK] Seq=75 Ack=78 Win=65536 Len=0 TSval=	25		TCP: 53409 → 25 [ACK] Seq=75 Ack=78 Win=65536 Len=0 TS...
112.350923073	C: DATA	25		SMTP: C: DATA
112.351233802	S: 354 End data with <CR><LF>.<CR><LF>	25		SMTP: S: 354 End data with <CR><LF>.<CR><LF>

Le résultat de **Analyse Suivre flux** sous wireshark renvoie ce que l'on a tapé lors du dialogue avec le Serveur SMTP comme le montre la figure suivante :





```

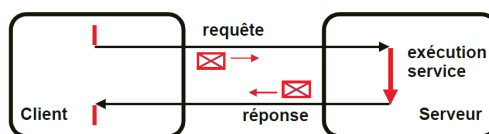
Wireshark · Suivre le flux TCP (tcp.stream eq 0) · Loopback: lo

220 PAF.lan ESMTP Postfix (Ubuntu)
HELO Bonjour Prof
250 PAF.lan
MAIL FROM:<Moi@NewYork.com>
250 2.1.0 Ok
RCPT TO:<pascal@PAF.lan>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Bonjour les L3 et Bonne Ann..e 2023 ;)
.
250 2.0.0 Ok: queued as DCFF3DE20CF
QUIT
221 2.0.0 Bye

```

— HTTP c'est du TCP un peu spécial... On est connecté et déconnecté..., on verra cela un peu plus tard !

Mais brièvement, à chaque page on est connecté, la page est téléchargée, quand elle est complètement chargée, le serveur nous déconnecte. C'est en cliquant sur un éventuel lien de la page que l'on sera à nouveau connecté.



3 TCP

C' est un mode **connecté** nécessitant que les hôtes mettent en place une **connexion** pour pouvoir échanger des données.

Ils se **synchronisent** les uns avec les autres pour gérer les **flux de paquets (stream)** et s'adaptent pour éviter une **congestion** réseau.

TCP fournit une vérification des erreurs grâce au contrôle fait dans le paquet IP : **Checksum**

3.1 La fiabilité

L'utilisation d'un principe appelé **PAR** pour **Positive Acknowledgment with Retransmission** ou **Accusé de réception positif avec retransmission** permet au protocole TCP de garantir des transmissions fiables !

Tout **Émetteur** utilisant PAR envoie à nouveau les données si le **Récepteur** distant ne renvoie pas un **ACK** précisant que les données sont bien arrivées.

L'unité utilisée pour l'échange de données entre client/serveur en TCP est appelés **segment**. Chaque segment contient une partie des données.

3.2 La segmentation

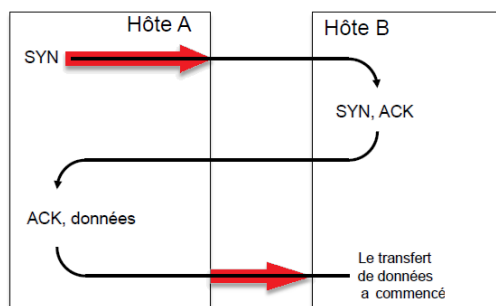
- L'unité utilisée pour l'échange de données entre Émetteur et Récepteur en TCP est appelée **segment**.
- Chaque segment contient **un total de contrôle** que le destinataire utilise pour vérifier que les données n'ont pas été altérées durant la transmission.
- Si le segment de données est reçu en parfait état, le récepteur renvoie un accusé de réception positif **ACK** à l'émetteur.
- Si c'est négatif, le récepteur élimine ce segment de données.
- Après un délai d'attente déterminé, l'émetteur retransmet le segment pour lequel aucun accusé de réception positif n'a été reçu.

3.3 Caractéristiques de TCP

- TCP est un protocole orienté connexion. Il établit donc une connexion en toute logique de bout en bout entre 2 hôtes ui doivent communiquer.

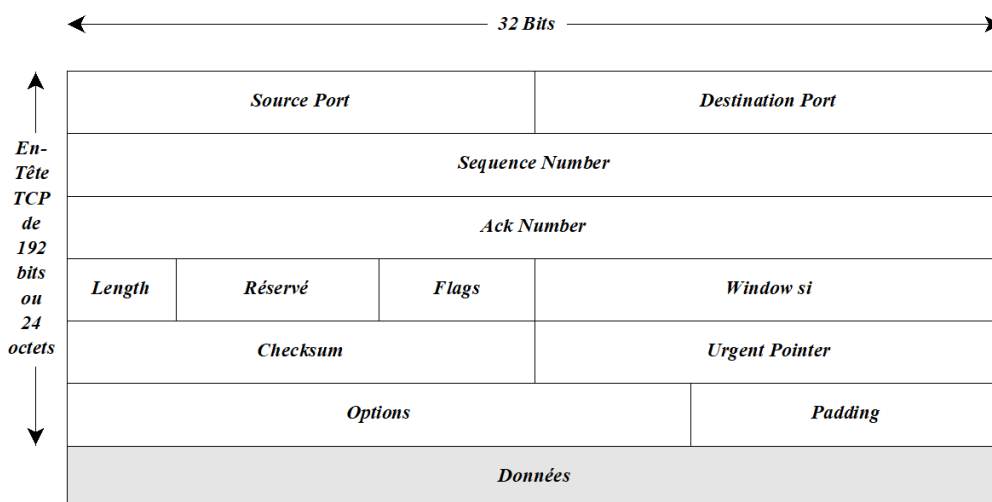


- Une information de contrôle, appelée **handshake** pour poignée de main (et non le poignet de la main ☺...) est échangée entre les 2 extrémités afin d'établir un dialogue avant la transmission des données.
- TCP utilise un mécanisme de **poignée de main** mettant en œuvre 3 échanges de segment : **three-way handshake** (c'est cool man ☺)



Voir plus loin : **Un exemple et une capture wireshark**

3.4 Entête TCP



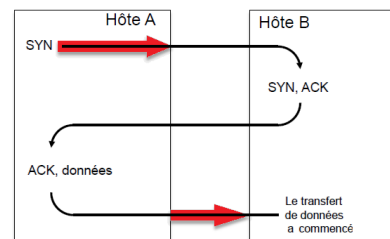
Les champs de l'entête TCP incluent

- Un champ **Source Port** correspondant N° du port appelant
- Un champ **Destination Port** correspondant N° du port appelé
- Des champs **Sequence Number** et **Acknowledgment Number** utilisés pour la **fiabilité** et éviter la **congestion**
- Un champ **Header Length** correspondant à la taille de l'entête
- Un champ **Reserved** réserve pour une utilisation future (j'adore le futur ...)
- Un champ **Flags** permettant de Contrôler l'en-tête TCP
- Un champ **Windows Size** qui correspond à la taille de la fenêtre
- Un champ **Checksum** est la somme de contrôle de l'entête et des champs utilisés, pour la vérification des erreurs

3.5 Établissement d'une connexion

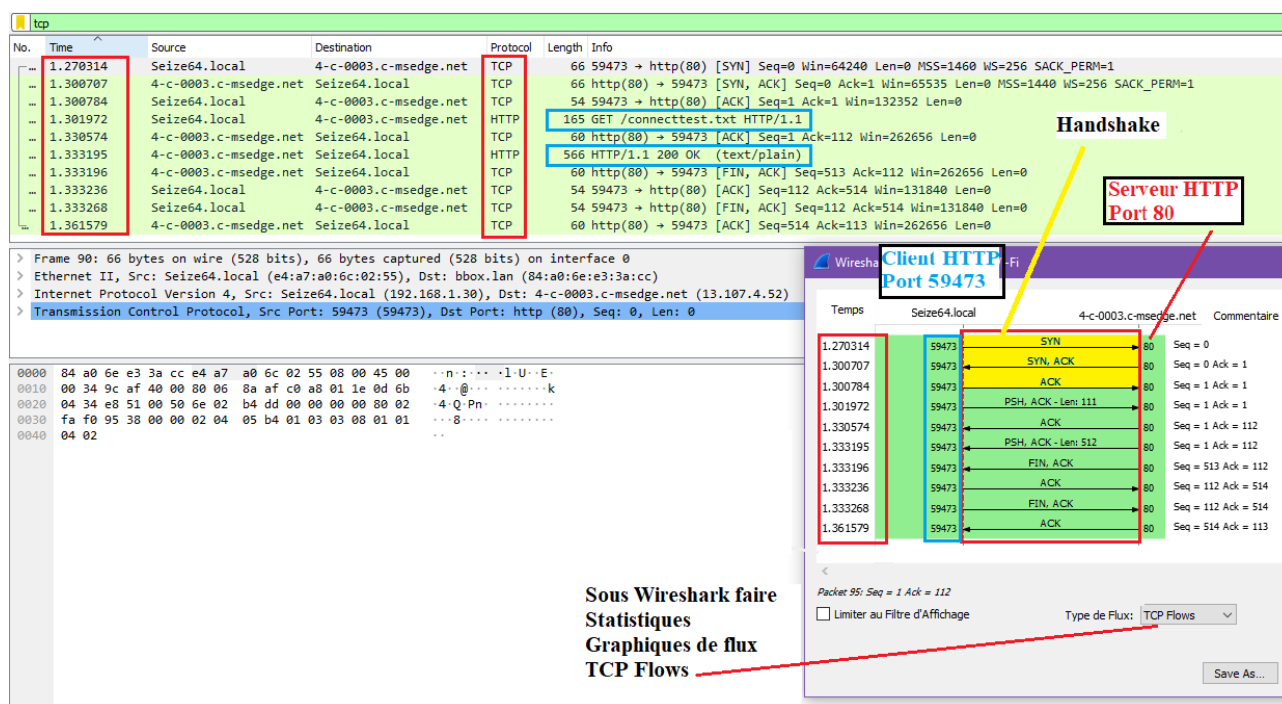
1. Un **serveur** ouvre une **socket** et se met en **attente passive de demandes de connexion** d'un client.
2. Un client effectue une **ouverture active** en 3 temps :
 - (a) Le client envoie un segment **SYN** au serveur,
 - (b) Le serveur répond par un segment **SYN/ACK**,
 - (c) Le client confirme par un segment **ACK**.
3. DATA
 - (a) Le client **PSH** les DATA et un **ACK** (HTTP ici !)
 - (b) Le serveur renvoie un accusé de réception **ACK**
 - (c) Le serveur envoie des **DATA** (la page Web HTTP)
4. Serveur envoie un **FIN** pour signaler qu'il a fini d'envoyer les données
5. Le client envoie un accusé de réception **ACK**
6. Le client envoie un **FIN** plus **ACK** pour signaler qu'il est d'accord
7. Le serveur envoie un accusé de réception **ACK**

Le **three-way handshake**



3.6 Un exemple et une capture wireshark

Graphique de flux TCP exemple CONNECTE en HTTP/TCP



4 UDP

Le protocole UDP permet aux applications d'accéder directement à un service de transmission de **datagrammes**, tel que le service de transmission qu'offre IP !

4.1 Caractéristiques de UDP

- Il possède un mécanisme permettant d'identifier les processus d'application grâce aux **N° de ports**
- Il est orienté **datagrammes**, pas de connexion, **permettant ainsi d'éviter les problèmes** liés à l'ouverture



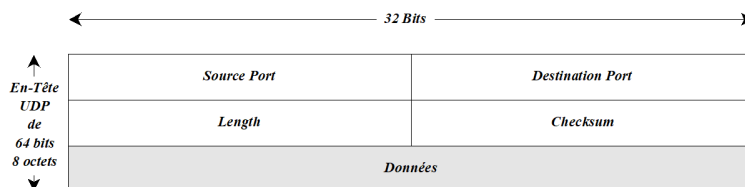
- le maintien
 - la fermeture
- des connexions, on n'est pas connecté donc c'est logique... ☺
- Il est **très performant pour les applications en diffusion/multidiffusion**. Les applications de type **interrogation-réponse (Telles DHCP et DNS)** peuvent utiliser ce protocole. La réponse peut être utilisée comme étant un accusé de réception positif à l'interrogation. Si pas de réponse au bout d'un certain intervalle de temps, l'application redemande. Le protocole DHCP que nous allons voir plus loin fonctionne sur ce principe !
 - Il ne séquence pas les données. La remise conforme des données n'est pas garantie !
 - Il peut éventuellement vérifier l'intégrité des données et seulement des données avec le **checksum**
 - Il est plus rapide, plus simple et plus efficace que TCP mais moins robuste pour ce qui est de la fiabilité.

Voici quelques protocoles **très utilisés** qui utilisent ce protocole UDP : DHCP, DNS, NFS, SNMP, TFTP.

UDP "n'est pas fiable" mais on l'utilise beaucoup et de plus en plus surtout dans les DATA Center avec les "VxLAN"

4.2 Entête UDP

Il est très simple et de taille fixe ! 8 octets



Les champs de l'entête UDP incluent

- Un champ **Source Port** correspondant N° du port appelant
- Un champ **Destination Port** correspondant N° du port appelé
- Un champ **Header Length** correspondant à la taille de l'entête
- Un champ **Header Length** correspondant à la taille de l'entête

4.3 Deux captures wireshark d'un paquet UDP

1. Marrant, la box TV de chez moi communique en UDP sur le WIFI, sûrement avec la box ?

C'est quoi ce port Source ? 57790... bizarre... ça complotte ?

Peut importe juste voir qu'il est possible d'avoir des données qui transitent sur un réseau en UDP et rien au dessus et ça marche !

No.	Time	Source	Destination	Protocol	Length	Info
2	0.007778	BouygatelTV-271011614017944	Seize64.local	UDP	312	57790 → 59523 Len=270
4	0.366417	Seize64.local	bbox.lan	DNS	86	Standard query 0x315f PTR 254.1.168.192.in-addr.arpa

> Frame 2: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface 0

> Ethernet II, Src: BouygatelTV-271011614017944 (a8:d3:f7:ed:d2:fb), Dst: Seize64.local (e4:a7:a0:6c:02:55)

> Internet Protocol Version 4, Src: BouygatelTV-271011614017944 (192.168.1.28), Dst: Seize64.local (192.168.1.30)

> User Datagram Protocol, Src Port: 57790 (57790), Dst Port: 59523 (59523)

> Data (270 bytes)

```

0000 e4 a7 a0 6c 02 55 a8 d3 f7 ed d2 fb 08 00 45 00 ...1.U...E.
0010 01 2a 49 51 40 00 00 11 6c e7 c0 a8 01 1c c0 a8 ...*IQ@:@:1.....
0020 01 1e e1 be e8 83 01 16 30 41 48 54 54 50 2f 31 .....0AHTTP/1.
0030 2e 31 20 32 30 30 20 4f 4b 0d 0a 55 53 4e 3a 20 ...1.200 O K..USN:
0040 75 75 69 64 3a 61 38 64 33 66 37 65 64 2d 64 32 ...uid:a8d 3f7ed-d2
0050 66 62 2d 30 30 66 36 2d 37 62 63 65 2d 66 62 63 ...fb-00f6-7bce-fbc
0060 39 39 30 34 33 35 35 34 32 3a 3a 75 70 6e 70 3a ...99843554 2:upnp:
0070 72 6f 6f 74 64 65 76 69 63 65 0d 0a 43 41 43 48 ...rootdevi ce..CAG
0080 45 2d 43 4f 4e 54 52 4f 4c 3a 20 6d 61 78 2d 61 ...E-CONTR0 L: max-a
0090 67 65 3d 31 38 30 30 0d 0a 45 58 54 3a 20 0d 0a ...ge=1800..EXT: ..
00a0 53 54 3a 20 75 70 6e 70 3a 72 6f 6f 74 64 65 76 ...ST: upnp:rootdev
00b0 69 63 65 0d 0a 4c 4f 43 41 54 49 4f 4e 3a 20 68 ...ice..LOC ATION: h
00c0 74 74 70 3a 2f 2f 31 39 32 2e 31 36 38 2e 31 2e ...http://19 2.168.1.
00d0 32 38 3a 35 30 30 30 2f 75 70 6e 70 2f 64 65 76 ...28:5000/ upnp/dev
00e0 2f 61 38 64 33 66 37 65 64 2d 64 32 66 62 2d 30 .../a8d3f7e-d-d2fb-0
00f0 30 66 36 2d 37 62 63 65 2d 66 62 63 39 39 38 34 ...0f6-7bce-fbc9984
0100 33 35 35 34 32 2f 64 65 73 63 0d 0a 53 45 52 56 ...35542/de sc..SERV
0110 45 52 3a 20 4c 69 6e 75 78 2f 33 2e 31 30 2e 34 ...ER: Linu x/3.10.4
0120 36 20 55 50 6e 50 2f 31 2e 30 20 43 6c 69 6e 67 ...6 UPnP/1 .0 Cling
0130 2f 32 2e 30 0d 0a 0d 0a .../2.0...

```

2. Le protocole VxLAN, celui des DATA CENTER




```

> Frame 402: 164 bytes on wire (1312 bits), 164 bytes captured (1312 bits) on interface 0
> Ethernet II, Src: CadmusCo_7b:1f:b7 (08:00:27:7b:1f:b7), Dst: CadmusCo_99:05:bc (08:00:27:99:05:bc)
> Internet Protocol Version 4, Src: 2.2.2.2, Dst: 1.1.1.1
> User Datagram Protocol, Src Port: 63939 (63939), Dst Port: 4789 (4789)
  Virtual eXtensible Local Area Network
    Flags: 0x0800, VLAN Network ID (VNI)
      0... .. = GBP Extension: Not defined
      .... .0.. .. = Don't Learn: False
      .... 1... .. = VLAN Network ID (VNI): True
      .... .. 0... = Policy Applied: False
      .000 .000 0.00 .000 = Reserved(R): False
      Group Policy ID: 0
      VLAN Network Identifier (VNI): 16641664
      Reserved: 0
> Ethernet II, Src: ca:02:39:8c:00:1c (ca:02:39:8c:00:1c), Dst: ca:01:5d:84:00:1c (ca:01:5d:84:00:1c)
> Internet Protocol Version 4, Src: 192.168.16.112, Dst: 192.168.16.111
> Internet Control Message Protocol

```

Question ?

C'est quoi cette capture wireshark où j'ai 2 fois de l'Ethernet, 2 fois de l'IP, c'est un beau "bordel" dans les DC ☺...

Non non, ne cherchez pas à comprendre davantage, juste pour vous persuader que **UDP est utilisé et n'est pas l'ancêtre de TCP**

5 Conclusion

TCP et UDP sont les 2 modes de "connexion" sur Internet.

- L'un TCP est dit fiable car il permet à la couche application de s'appuyer sur ce qu'il fait, **vérifier les données mais il est lent**. Ce n'est pas grave les utilisateurs sont encore plus lents ...
- Le second UDP est dit non fiable car **il ne vérifie pas que tout s'est bien passé**. Par contre il est beaucoup **plus rapide**. Je vous rassure il est assez fiable pour ce que nous allons faire avec !
- **Les paquets IP sont segmentés s'ils dépassent la taille du MTU, ce qui est le cas dans 99% des cas.**
- **Ils sont découpés et numérotés au départ, rassemblés en fonction de leur numéro à l'arrivée.**

UDP n'est ni plus ni moins utilisé que TCP et vice versa, on a besoin des 2 !!!

