TP n°6 : Étude des protocoles TCP et UDP

Durée : 2h

Objectif principal

Être capable de mettre en avant les fonctionnalités différentes des protocoles de transport **TCP** et **UDP**

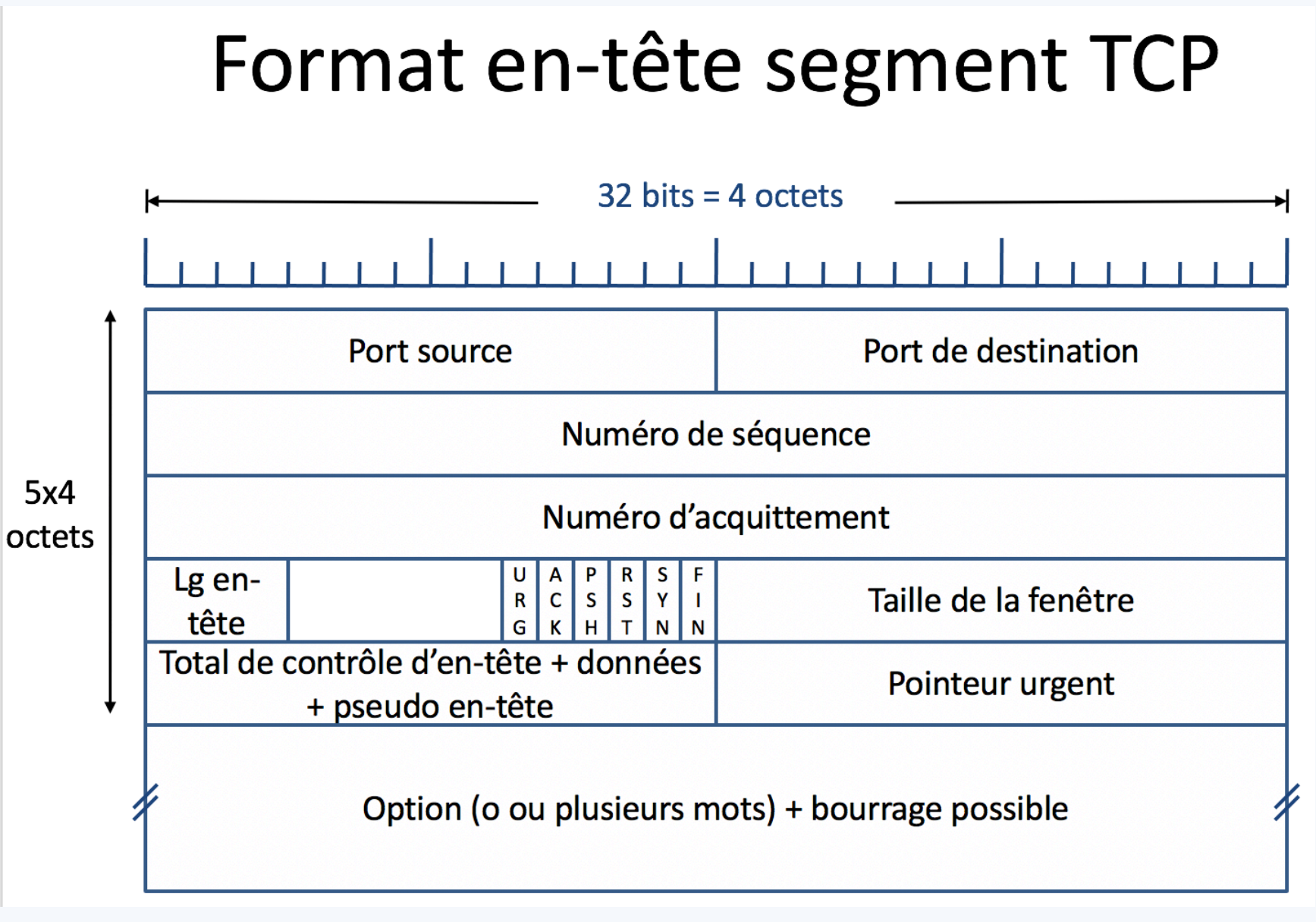
Protocole TCP - TRANSMISSION CONTROL PROTOCOL (Fiable)

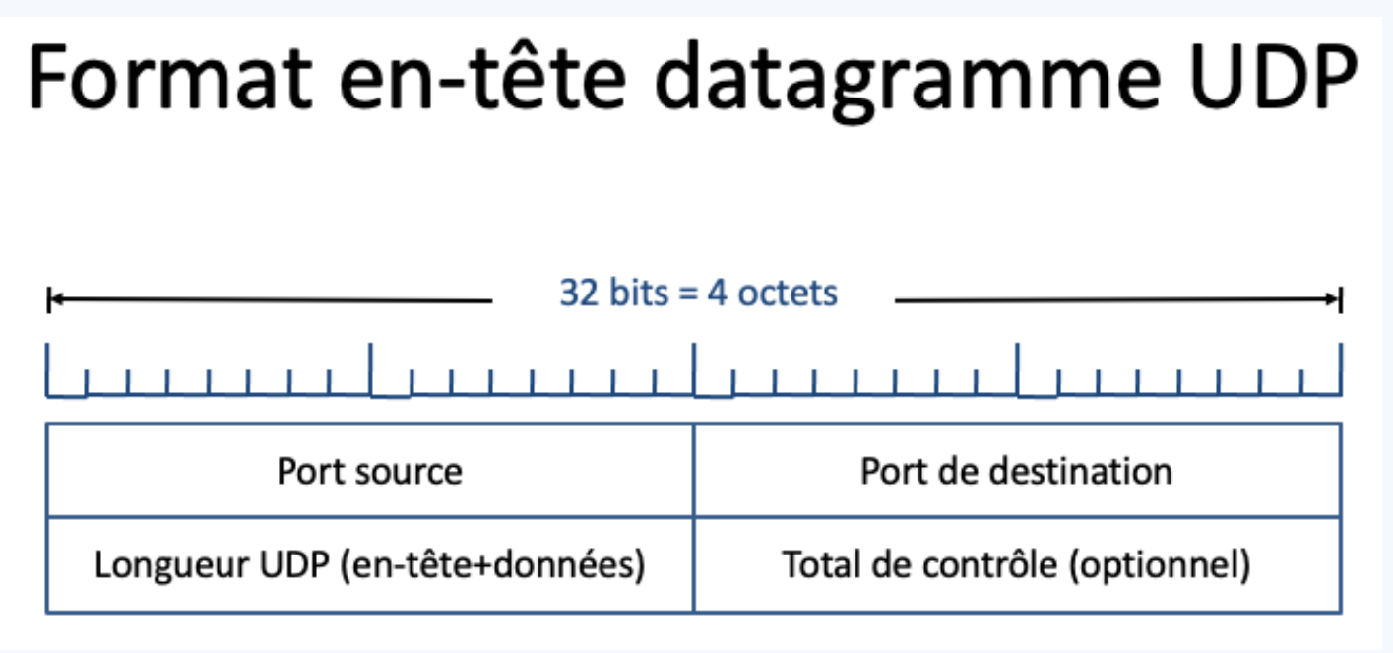
- Multiplexage d'applications
- Ré-émission de messages perdus et ré-ordonnancement
- Contrôle de flux
- Contrôle de congestion

Protocole UDP - USER DATAGRAM PROTOCOL (Rapide)

- Multiplexage d'applications

Rappels





Compte rendu individuel

Vous devez obligatoirement déposer chacun en fin de séance un compte rendu (CR) illustré de votre groupe au format **TP6.[pdf|odt|docx]** pour ce TP sur Moodle. Vous pouvez compléter ce CR après la séance en déposant un nouveau fichier **TP6v2.[pdf|odt|docx]**.

> Vos **CR seront vos uniques documents autorisés en CTP**. Il est donc très important de bien y faire figurer **tous les détails** (commandes précises, notions importantes, manips importantes, ...).

MÉMOS (Cheat Sheets)

Cours TCP/UDP | TCP/IP (source SANS) | Ports classiques (source packetlife.net)

Environnement de travail

Une machine virtuelle Linux dans un réseau interne derrière une passerelle de votre choix.

Étude des protocoles TCP et UDP

Le protocole **TCP est un protocole orienté connexion** : il y a donc une phase d'établissement de la connexion `(le "bonjour en 3 fois" "3-way handshake")` qui précède les échanges d'informations entre la source et le destinataire, cette phase d'échanges terminée la connexion se termine par une phase de fermeture de la connexion `(4 segments : FIN, ACK (pour fermer la connexion dans un sens), FIN, ACK (pour fermer la connexion dans l'autre sens))`.

Le protocole **UDP est un protocole non orienté connexion** : les informations à transmettre sont envoyées telles quelles, aucune phase d'établissement de la connexion, aucune fermeture.

Préambule - Rappels des services offerts par TCP et UDP -

À mettre dans votre compte rendu.

- Comment TCP exerce-t-il la ré-émission de messages perdus et le ré-ordonnancement ?
- Comment TCP exerce-t-il le contrôle de flux ?
- Comment TCP exerce-t-il le contrôle de congestion ?
- Comment TCP exerce-t-il le multiplexage des applications ?
- Parmi les services précédents, quels sont ceux offerts par UDP ?

Rappels Pratiques à l'IUT

> Vérifiez votre quota disque avec la commande Linux **quota**

> Pour voir la taille des répertoires, utilisez la commande **du --max-depth=1** (videz votre poubelle régulièrement).

> Pour **Wireshark**, **utilisez un filtre de capture** pour limiter les problèmes de blocage de machines virtuelles dues au trafic.

> Au cas où, vous pouvez relancer un environnement gnome "gelé" avec la commande **killall -3 gnome-shell**.

Étude du trafic TCP

Trafic normal

TCP est encapsulé dans SSH, FTP ou HTTP. Pour générer un trafic TCP vous pouvez donc utiliser ses services. Mais on peut également utiliser l'outil netcat permettant d'ouvrir des connexions réseau TCP ou UDP ce que vous allez d'abord utiliser ici.

Netcat permet d'ouvrir simplement un serveur grâce à la commande **nc -l hote num_port** : un serveur sur l'hôte "hote" à l'écoute sur le port "num_port". On peut aussi ne pas préciser l'hôte et l'ouverture se fera sur la boucle locale (interface lo).

Un client peut alors se connecter grâce à la commande **nc hote num_port**.

- Lancez Wireshark sur l'interface boucle locale (lo) avec un filtre de capture TCP
- Lancez dans un terminal un serveur sur la boucle locale à l'écoute sur le port 8000

```
nc -l 8000
```

- Lancez dans un autre terminal un client

```
nc localhost 8000
```

  Vous disposez alors d'un mini "chat" (une messagerie instantannée fonctionnant sur TCP) entre le client et le serveur. Vous tapez votre texte dans les terminaux du client et du serveur.

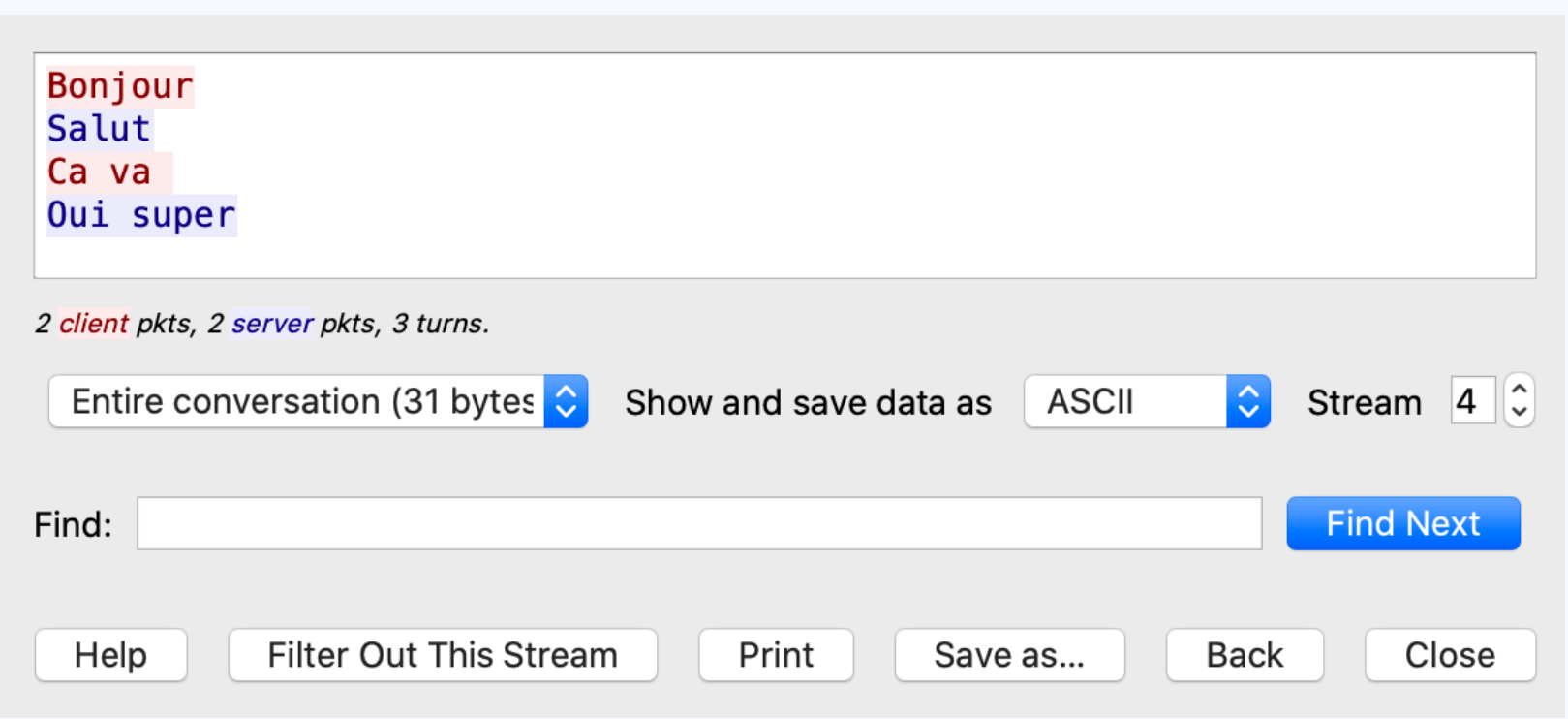
> Vous pouvez voir ce nouveau port ouvert avec la commande **ss -n | grep 8000** (ou **ss -lnp | grep 8000**). Remarque la commande netstat ne doit plus être utilisée comme par exemple les commandes ifconfig, arp, route, ... du paquet ("package") NET-TOOLS. Il faut utiliser à la place les commandes du paquet IPROUTE2 (comme par exemple ip, ss, ...) : IP COMMANDS.

- Effectuez un échange de mots

Par exemple : Bonjour | Salut | Ca va | Oui super

- Finissez la discussion par **ctrl+D**.
- Coupez la capture Wireshark et filtrez cette discussion.

En faisant un **clic droit** sur une trame de cet échange puis **"follow" | "TCP stream" (suivre ce flux TCP)**, vous pouvez voir par exemple une capture du flux TCP associé à l'exemple précédent.



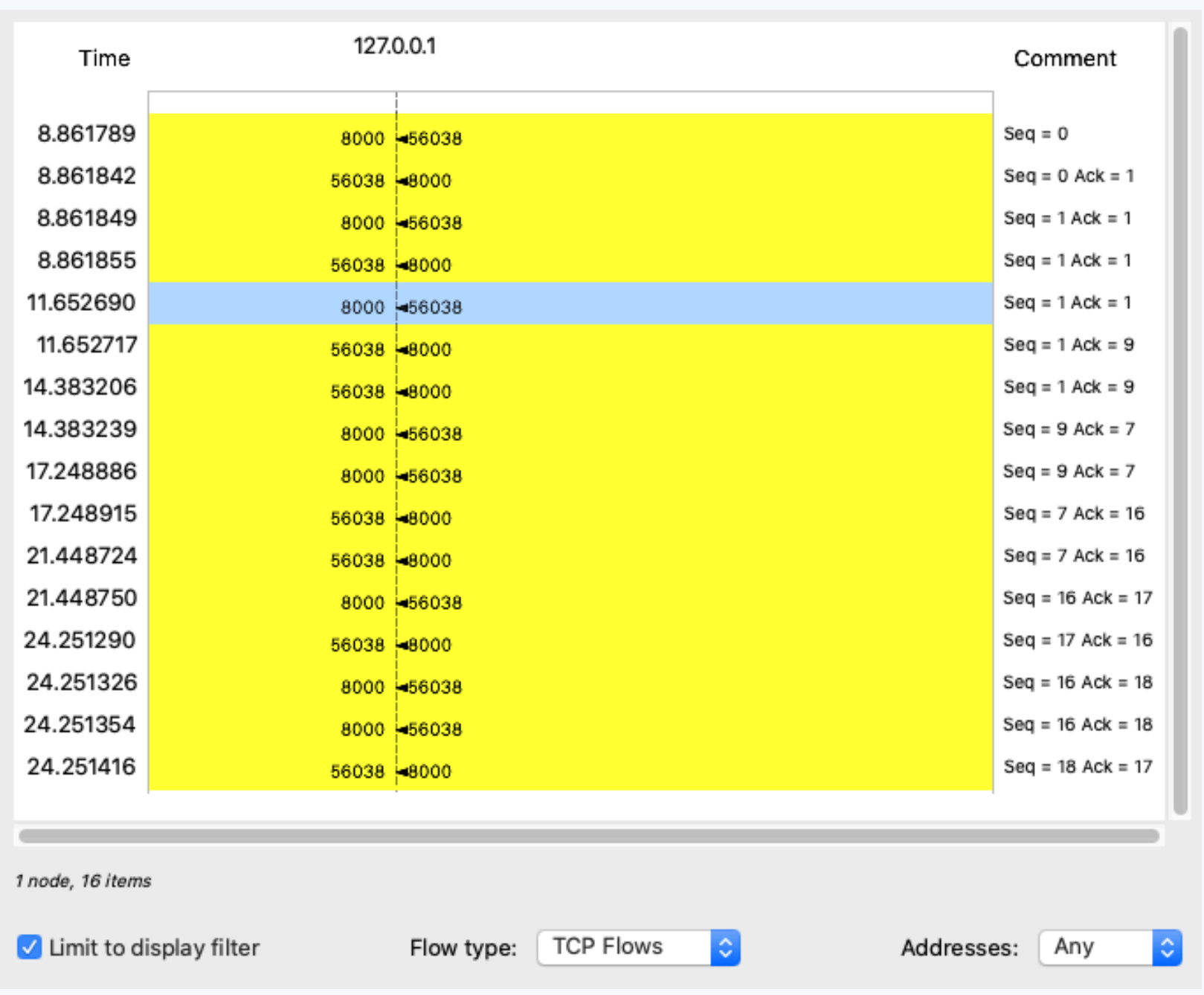
```
Bonjour
Salut
Ca va
Oui super
```

2 client pkts, 2 server pkts, 3 turns.

Entire conversation (31 bytes | Show and save data as: ASCII | Stream: 4

Find:

Dans le menu **"statistics"**, vous avez également **"Flow graph"** pour voir l'illustration de l'échange des trames. Spécifiez **"TCP flows" (flux TCP)** dans Flow type et **"Limit to display filter"** pour avoir que le trafic TCP concerné, et voir par exemple un graphe d'échanges TCP comme le suivant.



Time	127.0.0.1	Comment
8.861789	8000 ← 56038	Seq = 0
8.861842	56038 ← 8000	Seq = 0 Ack = 1
8.861849	8000 ← 56038	Seq = 1 Ack = 1
8.861855	56038 ← 8000	Seq = 1 Ack = 1
11.652690	8000 ← 56038	Seq = 1 Ack = 1
11.652717	56038 ← 8000	Seq = 1 Ack = 9
14.383206	56038 ← 8000	Seq = 1 Ack = 9
14.383239	8000 ← 56038	Seq = 9 Ack = 7
17.248886	8000 ← 56038	Seq = 9 Ack = 7
17.248915	56038 ← 8000	Seq = 7 Ack = 16
21.448724	56038 ← 8000	Seq = 7 Ack = 16
21.448750	8000 ← 56038	Seq = 16 Ack = 17
24.251290	56038 ← 8000	Seq = 17 Ack = 16
24.251326	8000 ← 56038	Seq = 16 Ack = 18
24.251354	8000 ← 56038	Seq = 16 Ack = 18
24.251416	56038 ← 8000	Seq = 18 Ack = 17

1 node, 16 items

☑ Limit to display filter | Flow type: TCP Flows | Addresses: Any

- Vérifiez que les numéros de séquence et d'acquittement correspondent bien à ce que vous avez vu en cours et TD.

Par exemple : numéro de séquence (Seq) + longueur des données (Len) = numéro d'acquittement (Ack)

- Le champ taille de fenêtre (Win) est codé sur 2 octets, sa valeur maximale est donc de 65535 octets. Pourtant vous pouvez apercevoir des valeurs plus grande pour ce champ dans cet échange.

Vous pouvez avoir cela par exemple

```
127.0.0.1          127.0.0.1          TCP          62  8000 → 56038 [PSH, ACK] Seq=1 Ack=9 Win=408256 Len=6
```

  Recherchez dans l'analyse Wireshark d'une de ces trames, une raison grâce à laquelle c'est possible.
- Identifiez les segments TCP liés à la phase d'établissement de la connexion TCP ("3-way handshake") et vérifiez - par rapport au cours - les fanions/drapeaux/flags TCP apparaissant ainsi que les numéros de séquence et d'acquittement.

> Attention, par défaut wireshark indique des numéros de séquence relatifs (Le premier numéro de séquence est décalé à 0). Pour voir les vrais numéros de séquence, il faut aller dans le menu préférences de wireshark, puis dans le protocole TCP, vous décochez la case "Relative sequence numbers". Remettez ensuite l'option pour avoir des numéros de séquence et d'acquittement relatifs (plus facile pour étudier TCP)

- Identifiez les segments TCP liés à la fermeture de la connexion TCP et vérifiez - par rapport au cours - les fanions/drapeaux/flags TCP apparaissant.

Connexions multiples

Ouvrez maintenant plusieurs connexions SSH sur une même machine. Les connexions se mélangent-elles ? Pourquoi ?

Consultez un même site depuis plusieurs onglets d'un même navigateur web. Les connexions se mélangent-elles ? Pourquoi ?

Une connexion est identifiée par un couple <adresse ip, numéro de port> appelé **socket** (une "prise" en français)

Ouverture vers un port non utilisé

- Capturez les trames TCP issues d'une demande de connexion vers un port non utilisé.

Faites par exemple un ssh vers une deuxième machine où le service ssh est fermé, ou alors utilisez netcat en ouvrant un client sur un port fermé (serveur non actif sur ce port).

- Expliquez ce qu'il se passe.

Connexion coupée

- Capturez les trames TCP issues d'une connexion coupée.

Faire un ssh vers une deuxième machine, et une fois la connexion établie, coupez le réseau de cette deuxième machine (déconnectez en mode graphique ou débranchez le câble réseau).

- Que se passe-t-il sur le réseau ? Mettre une capture de ces segments dans votre compte-rendu à partir de l'outil de visualisation d'échanges de Wireshark.
- Le timer de ré-émission utilisé dans ce cas par TCP, a-t-il une durée fixe ? Comment évolue-t-il ? Quel intérêt ?
- Combien de temps TCP insiste-il avant d'arrêter de réémettre.
- Si la deuxième machine se reconnecte, que se passe-t-il ?

Analyse de ports

Une **analyse de ports** ou **scan de ports** ou **balayage de ports** consiste à trouver les ports ouverts d'une machine.

Avec Netcat

Il est possible de faire avec netcat et l'option -z

- Lancez dans un terminal un serveur sur la boucle locale à l'écoute sur le port 800

```
nc -l 800
```

- Lancez Wireshark sur l'interface boucle locale (lo)
- Lancez dans un terminal le scan Netcat sur la plage réduite 800-802

```
nc -z localhost 800-802
```

- Coupez la capture Wireshark et filtrez cette discussion.

> Filtre Wireshark pour aider **tcp.port == 800**

- Que fait netcat pour scanner un port ? Mettre une capture du scan du port 800 dans votre compte-rendu. Mettre aussi une capture du scan d'un port non ouvert comme le 801 par exemple dans votre compte-rendu.

Avec Nmap

Un outil bien plus puissant et tout aussi connu pour faire de l'analyse de ports est Nmap. Vous allez pour finir ce TP capturer et analyser grâce à Wireshark différent type de scans via Nmap.

Les scans se font sur votre localhost **- Attention : il n'est pas autorisé de scanner une machine qui n'est pas à vous -**

- Pour chacun de ces scans, vous devez comme l'analyse du scan via netcat :
  1. Lancez un serveur dans un terminal (**nc -l 800**)
  2. Lancez Wireshark sur la boucle locale, le filtre d'affichage **tcp.port == 800** va vous aider
  3. Lancez le scan puis analyser le résulat : quel(s) format(s) de segment TCP est/sont envoyé(s) pour chaque type de scan ? Mettre les captures de chaque scan du port 800 dans votre compte-rendu.

Les scan à analyser sont les suivants :

1. **nmap -sS localhost**
2. **nmap -sT localhost**
3. **nmap -sA localhost**
4. **nmap -sF localhost**
5. **nmap -sM localhost**
6. **nmap -sN localhost**
7. **nmap -sW localhost**
8. **nmap -sX localhost**

Pour finir, Nmap peut aussi faire des scans UDP avec la commande **nmap -sU localhost**.

- Lancez Wireshark. Lancez la commande **nmap -sU localhost**. Filtrez l'analyse d'un port, par exemple **udp.port == 800**. Conclure sur le fonctionnement de ce scan : quel type de trame est envoyé ? Quel type de de trame est retourné ? Mettre une capture dans votre compte-rendu.

N'oubliez pas de déposer votre compte rendu.

SYNTHÈSE

À l'issue de ce TP :

- Vous avez pu vérifier les fonctionnalités de TCP et UDP

Protocole TCP - TRANSMISSION CONTROL PROTOCOL (Fiable)

- Multiplexage d'applications
- Ré-émission de messages perdus et ré-ordonnancement
- Contrôle de flux
- Contrôle de congestion

Protocole UDP - USER DATAGRAM PROTOCOL (Rapide)

- Multiplexage d'applications

- Vous avez également pu voir, si vous ne les connaissiez pas, deux utilitaires réseau pratiques : un utilitaire d'exploration de réseau **nmap** et un utilitaire **netcat** permettant d'ouvrir des connexions réseau TCP, et lancer des datagrammes UDP.

(Les commandes font également l'objet de questions aux DS et évaluations écrites)