

*Département de génie informatique et génie logiciel*

**INF3405**

Réseaux informatiques

**TP3 :**

**Analyse d’applications client-serveur avec WireShark**

Soumis par :

**Grégoire Catimel-Marchand – 1932276**

**Simon Gagné - 1897255**

Section 3

**PARTIE 1**

**Question 1) Quel filtre appliqueriez-vous afin d’afficher uniquement les échanges entre le client et le serveur?**

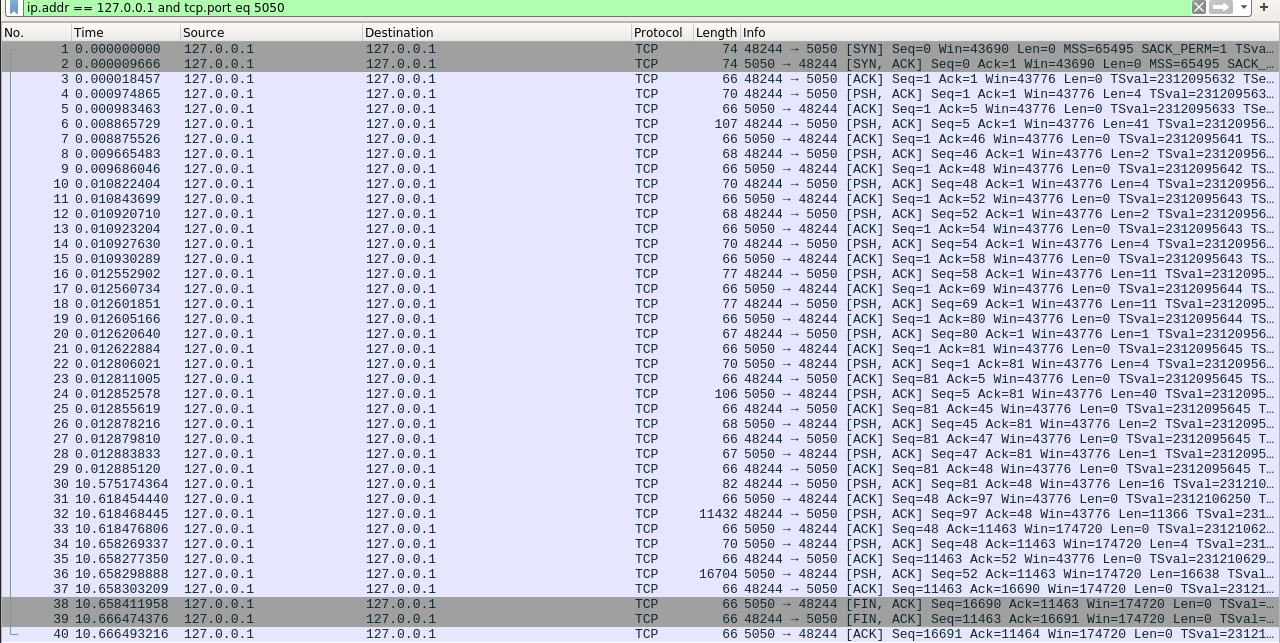
Le filtre :

ip.addr == 127.0.0.1 and tcp.port == 5050

Permet d’afficher uniquement les échanges entre l’IP source 127.0.0.1 et l’IP destination 127.0.0.1 utilisant le port 5050 soit l’IP et le port utilisé par l’application client et serveur.

**Question 2) À la lumière de vos observations, dites quel protocole de la couche 4 est utilisé pour la communication entre le client et le serveur.**

La couche 4 est la couche de transport. À l’aide du filtre, il est possible de voir que tous les échanges ont eu lieu avec le protocole TCP (Transmission control protocol).



**Question 3) Combien de paquets et d’octets de données ont été envoyés du client vers le serveur et du serveur vers le client?**

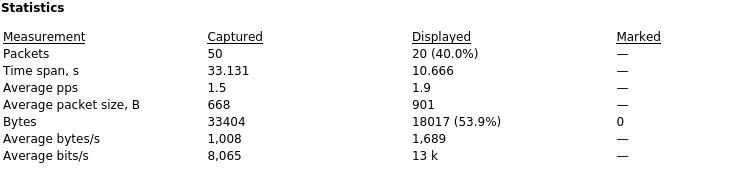
En se fiant au premier paquet envoyé, on peut déterminer que :

client → serveur == 48244 → 5050

**Pour le serveur :**

En utilisant le filtre : ip.addr == 127.0.0.1 and tcp.srcport == 5050

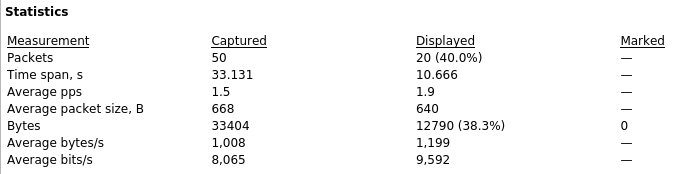
On peut donc voir que :

Le serveur a envoyé 20 paquets pour un total de 18017 octets.

**Pour le client :**

En utilisant le filtre : ip.addr == 127.0.0.1 and tcp.srcport == 48244

On peut voir que :

Le client a envoyé 20 paquets pour un total de 12790 octets.

**Question 4) Normalement, le standard IEEE 802.3 limite la taille d’une trame Ethernet à 1518 octets. Dans votre capture Wireshark, existe-t-il des paquets ayant une taille supérieure à 1518 octets? Si oui, expliquez pourquoi et comment ce paquet réussit à transiger sur le réseau alors que sa taille est plus grande que celle spécifiée par le standard.**

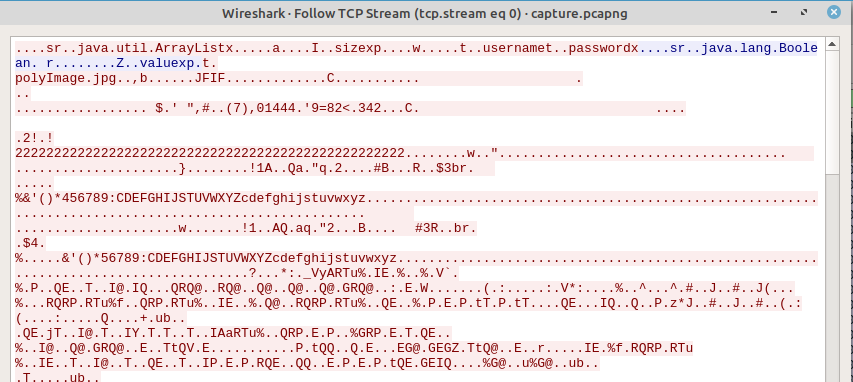
Il existe 2 paquets dépassant cette limite :



Le protocole IPv6 permet l’envoi de paquets jusqu’à 65535 octets grâce à certaines modifications au protocole TCP. Plus précisément, en ajoutant une extension à l’entête existante.

Parce que la longueur d’une trame n’est pas spécifiée dans un entête TCP, rien ne limite la longueur de celle-ci. Cependant, lors de la connection avec le protocole IPv6 la taille est limitée a 65535 octets sauf dans le cas ou on utilise des jubograms.

**Question 5) Quel type d’information êtes-vous capables d’extraire de Wireshark en lien avec l’authentification au serveur de traitement d’images?**

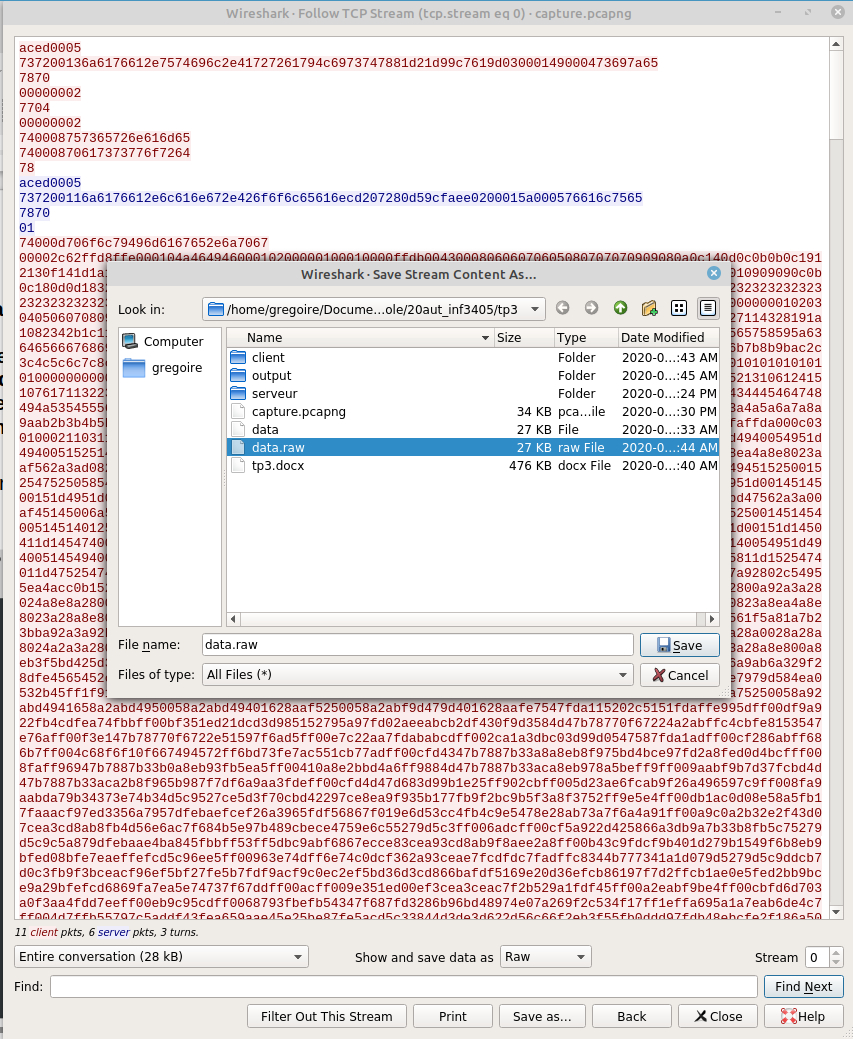


On peut facilement observer le nom de l’utilisateur soit « username » et son mot de passe soit « password » au début de la capture suivante.

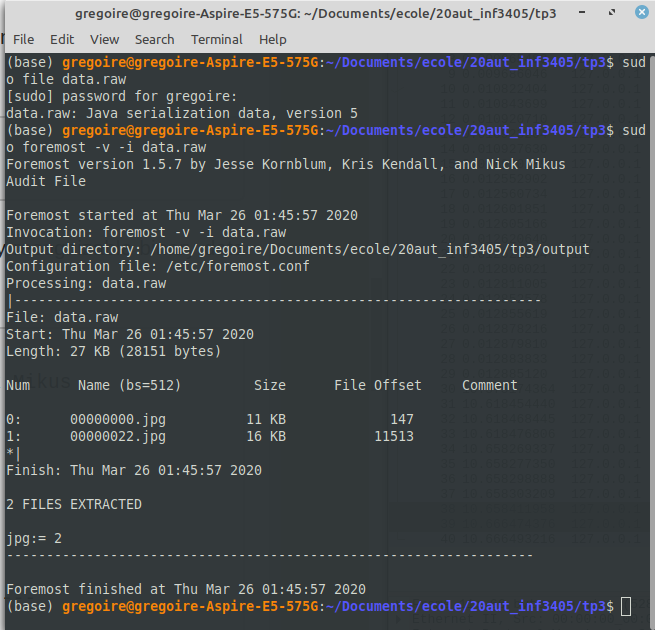
**Question 6) Il est possible, avec Wireshark, d’extraire l’image envoyée par le client ou l’image traitée. Donnez les étapes à suivre, incluant des captures d’écran montrant chaque étape permettant l’extraction de l’image envoyée du client vers le serveur. Servez-vous des propriétés du fichier.jpg énoncées plus haut. Indice: utilisez le programme WinHex après avoir sauvegardé le flot de données en format " Raw ”**

Puisque WinHex n’est pas disponible pour Linux, j’ai utilisé Foremost 1.5.7 après avoir avoir préalablement demandé et approuvé avec professeur Quintero.

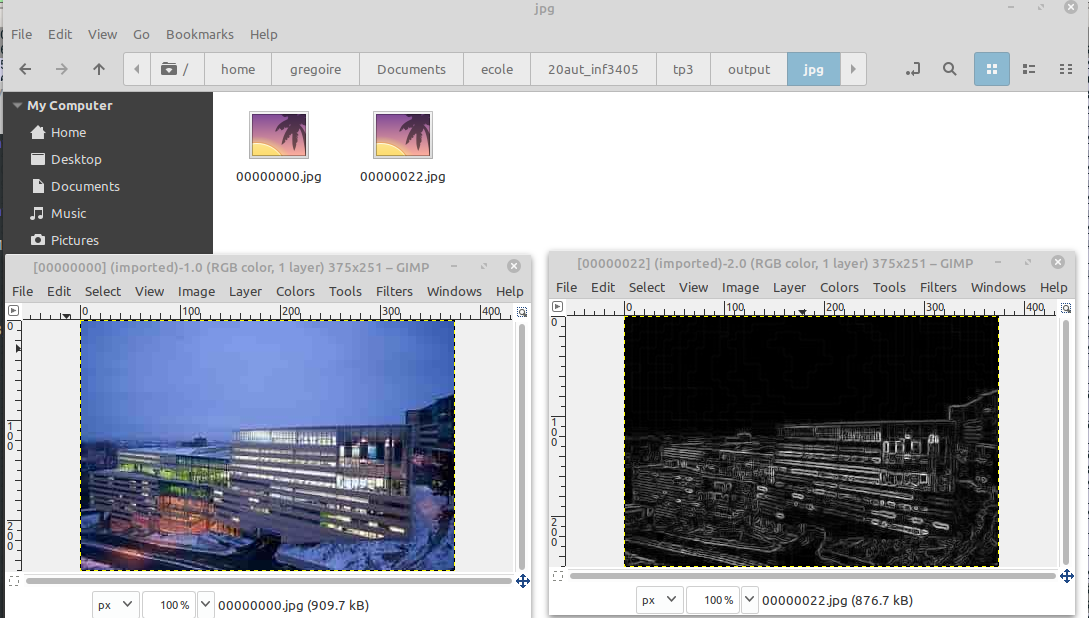
Étape 1 : On sauvegarde les données en format raw.

****

Étape 2 : La comande « foremost -v -i data.raw » permet d’extraire les données d’une capture dans un environnement Linux.



Étape 3 : On peut ouvrir les photos dans le output de la commande foremost.



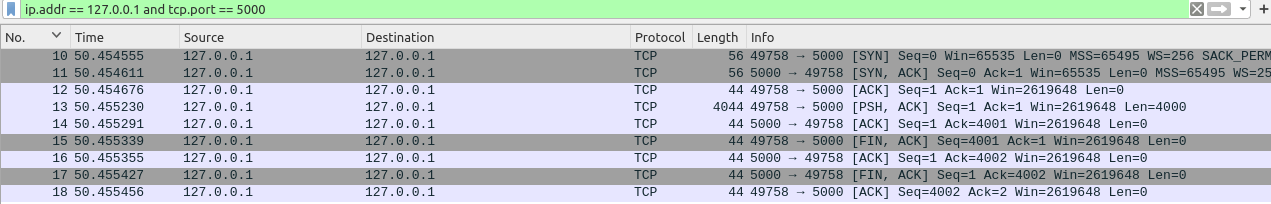
**Question 7) Suite à toute cette analyse que pouvez-vous conclure quant à la sécurité de l’application de traitement d’images que vous avez développé lors du travail pratique no.2**

Cette application est très peu sécuritaire, il est possible d’en extraire toutes données d’identification (nom d’utilisateur et mot de passe) ainsi que toutes données traités (images).

**PARTIE 2**

**Mode 1 :**

**Question 1) Quel protocole de la couche transport est utilisé? Dans le cas de TCP, montrer le tout premier échange entre le client et le serveur lors de l’initialisation de la connexion, comment ce nomme cet échange? Dans le cas d'UDP, est-ce que ce même échange à lieu? Pourquoi?**

Le protocole est TCP.

La connexion est établie par une phase appelée le «three-way handshake». Cette phase en trois étapes permet de synchroniser 2 appareils.

Étape 1 : le client tente d’établir une connexion avec le serveur et lui envoie un paquet.

* Le SYN flag est a 1 désignant une synchronisation pour une tentative de connexion.
* Le numéro de paquet est 0 (seq = 0) puisque c’est le premier paquet envoyé par le client.
* Voir No. 10 sur la capture d’écran Wireshark.

Étape 2 : le serveur ayant reçu la demande de connexion du client, il lui envoie un paquet.

* Le SYN flag est a 1 désignant une tentative de connexion et le ACK flag est aussi a 1 désigrant un acusé de réception.
* Le numéro de paquet est 0 (seq = 0) puisque c’est le premier paquet envoyé par le serveur.
* Voir No. 11 sur la capture d’écran Wireshark.

Étape 3 : le client ayant reçu une réponse positive pour la connexion au serveur lui envoie un paquet.

* Le ACK flag est a 1 pour désigner avoir reçu le paquet précédent.
* Le numéro de paquet est 1 (seq = 1) puisque c’est le deuxième paquet envoyé par le client
* Voir No. 12 sur la capture d’écran Wireshark.

**Question 2) En vous basant sur les informations recueillies par Wireshark, indiquez les ports source et destination utilisés par la couche 4.**

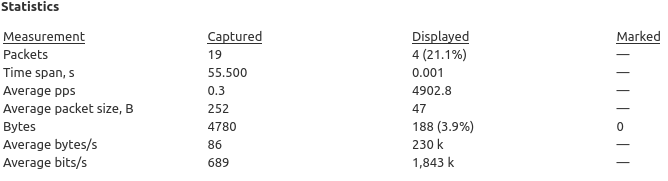
Il est possible de voir sur la capture d’écran que le port source est 49758 et le port destination est 5000 car le premier échange a lieu de la source vers la destination avec le protocole TCP.

**Question 3) Combien de paquets et d’octets contenant des données ont été envoyés par le client vers le serveur? Par le serveur vers le client? Montrer où vous avez trouvé cette information.**

**Pour le serveur :**

En utilisant le filtre : ip.addr == 127.0.0.1 and tcp.srcport == 5000

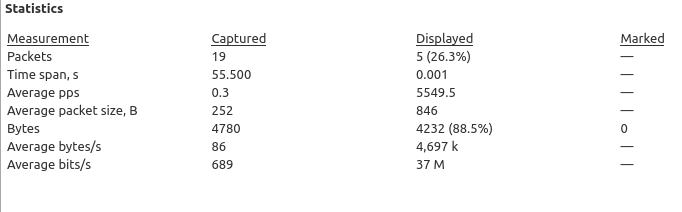
On peut donc voir que :

Le serveur a envoyé 4 paquets pour un total de 188 octets.

**Pour le client :**

En utilisant le filtre : ip.addr == 127.0.0.1 and tcp.srcport == 49758

On peut voir que :



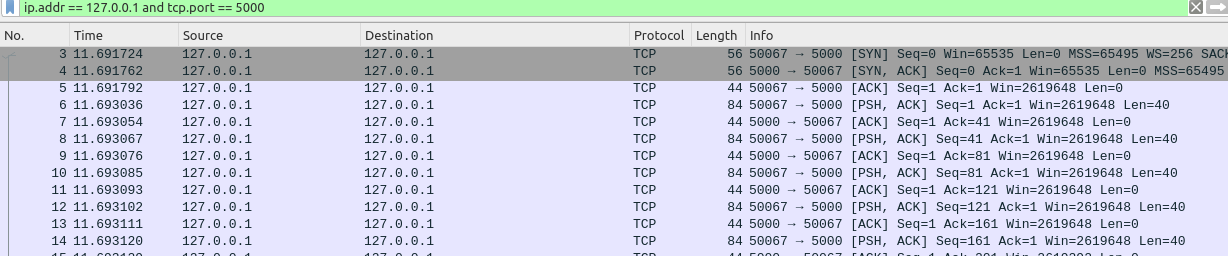
Le client a envoyé 5 paquets pour un total de 4232 octets.

**Question 4) À la lumière de votre analyse, que fait le client? Selon vous, combien d’itérations le client a-t-il faites pour envoyer ces données?**

Le client semble envoyer des données au serveur. Considérant que 4 de ces paquets sont des paquets de synchronisation, il y a eu 1 itération.

**Mode 2 :**

**1) Quel protocole de la couche transport est utilisé? Dans le cas de TCP, montrer le tout premier échange entre le client et le serveur lors de l’initialisation de la connexion, comment ce nomme cet échange? Dans le cas d'UDP, est-ce que ce même échange à lieu? Pourquoi?**

Le protocole est TCP. La description du tout premier échange est la même que pour le mode 1.

**2) En vous basant sur les informations recueillies par Wireshark, indiquez les ports source et destination utilisés par la couche 4.**

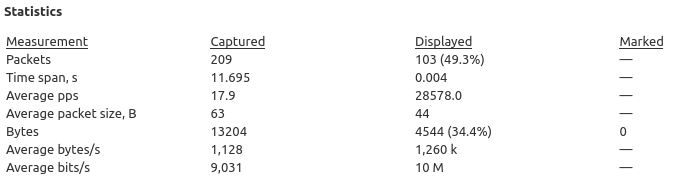
Il est possible de voir sur la capture d’écran que le port source est 50067 et le port destination est 5000 car le premier échange a lieu de la source vers la destination avec le protocole TCP.

**3) Combien de paquets et d’octets contenant des données ont été envoyés par le client vers le serveur? Par le serveur vers le client? Montrer où vous avez trouvé cette information.**

**Pour le serveur :**

En utilisant le filtre : ip.addr == 127.0.0.1 and tcp.srcport == 5000

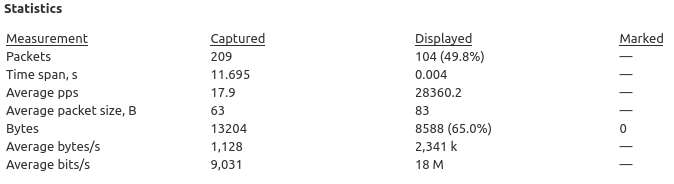
On peut donc voir que :

Le serveur a envoyé 103 paquets pour un total de 4544 octets.

**Pour le client :**

En utilisant le filtre : ip.addr == 127.0.0.1 and tcp.srcport == 50067

On peut voir que :

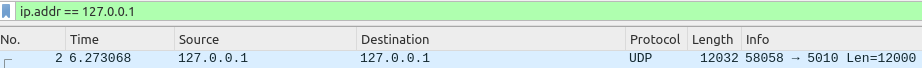
Le client a envoyé 104 paquets pour un total de 8588 octets.

**4) À la lumière de votre analyse, que fait le client? Selon vous, combien d’itérations le client a-t-il faites pour envoyer ces données?**

Le client semble envoyer des données au serveur. Considérant que 4 de ces paquets sont des paquets de synchronisation, il y a eu 100 itérations.

**Mode 3 :**

**1) Quel protocole de la couche transport est utilisé? Dans le cas de TCP, montrer le tout premier échange entre le client et le serveur lors de l’initialisation de la connexion, comment ce nomme cet échange? Dans le cas d'UDP, est-ce que ce même échange à lieu? Pourquoi?**

Le protocole est UDP. Contrairement à TCP, UDP ne requiert aucune communication préalable pour établir une connexion. Ainsi UDP communique sans établir de connexion.

**2) En vous basant sur les informations recueillies par Wireshark, indiquez les ports source et destination utilisés par la couche 4.**

Il est possible de voir sur la capture d’écran que le port source est 58058 et le port destination est 5000 car l’échange a lieu de la source vers la destination avec le protocole UDP.

**3) Combien de paquets et d’octets contenant des données ont été envoyés par le client vers le serveur? Par le serveur vers le client? Montrer où vous avez trouvé cette information.**

Un seul paquet de 12032 octets a été envoyé du client ver le serveur. Ces informations sont visibles sur la capture d’écran plus haut.

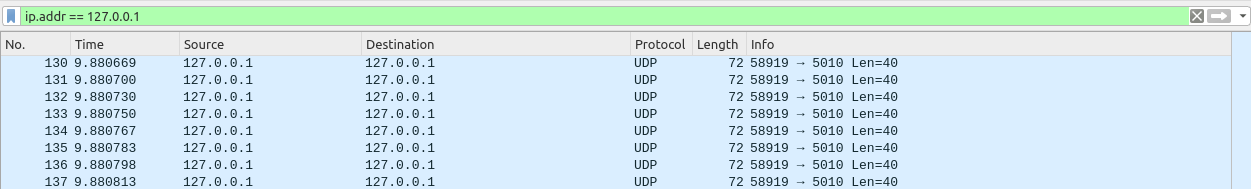
**4) À la lumière de votre analyse, que fait le client? Selon vous, combien d’itérations le client a-t-il faites pour envoyer ces données?**

Le client envoie des données au serveur. Une seule itération a été faite pour envoyer ces données.

**Mode 4 :**

**1) Quel protocole de la couche transport est utilisé? Dans le cas de TCP, montrer le tout premier échange entre le client et le serveur lors de l’initialisation de la connexion, comment ce nomme cet échange? Dans le cas d'UDP, est-ce que ce même échange à lieu? Pourquoi?**

Le protocole est UDP. Contrairement à TCP, UDP ne requiert aucune communication préalable pour établir une connexion. Ainsi UDP communique sans établir de connexion. La description du modèle UDP est la même que pour le modèle 3.



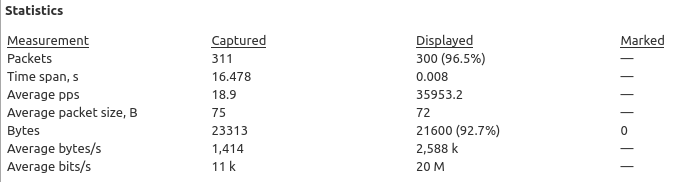
**2) En vous basant sur les informations recueillies par Wireshark, indiquez les ports source et destination utilisés par la couche 4.**

Il est possible de voir sur la capture d’écran que le port source est 58919 et le port destination est 5000 car l’échange a lieu de la source vers la destination avec le protocole UDP.

**3) Combien de paquets et d’octets contenant des données ont été envoyés par le client vers le serveur? Par le serveur vers le client? Montrer où vous avez trouvé cette information.**

En utilisant le filtre : ip.addr == 127.0.0.1 and udp == 58919

On peut donc voir que :

Le client a envoyé 300 paquets pour un total de 21600 octets.

Le serveur n’a rien envoyé.

**4) À la lumière de votre analyse, que fait le client? Selon vous, combien d’itérations le client a-t-il faites pour envoyer ces données?**

Le client envoie des donées au serveur. Le client a fait 300 itérations puisqu’il y a 300 paquets.

**Partie 3 : Analyse des performances et protocole TCP**

**Question 1) Comparez la performance des envois de données pour le mode 1 et le mode 2. Qu’est-ce qui diffère entre ces deux modes? Lequel est le plus performant selon vous et pourquoi?**

Le premier mode envoie 1 paquet d’une longueur de 4000 octets.

Le deuxième mode envoie 100 paquets d’une longueur de 40 octets.

Dans les deux cas, un paquet a un entête de 44 octets.

Pour déterminer le mode le plus efficace, calculons le pourcentage de données utiles envoyé par chaque mode.

Mode 1 :

% données utiles = (# de trames \* données) / (# de trames \* (données + entête))

= 4000 / 4044

= 98.9% de données utiles

Mode 2 :

% données utiles = (# de trames \* données) / (# de trames \* (données + entête))

= (100 \* 40) / (100\* (40 + 44))

= 47.6% de données utiles

Le mode 1 est plus performant, il gaspille moins de bande passante en envoyant 98.9% de données utiles.

**Question 2) Comparer la performance des envois de données pour le mode 3 et le mode 4. Qu’est-ce qui diffère entre ces deux modes? Lequel est le plus performant selon vous et pourquoi?**

Le troisième mode envoie 1 paquet d’une longueur de 12000 octets.

Le deuxième mode envoie 300 paquets d’une longueur de 40 octets.

Dans les deux cas, un paquet a un entête de 32 octets.

Pour déterminer le mode le plus efficace, calculons le pourcentage de données utiles envoyé par chaque mode.

Mode 3 :

% données utiles = (# de trames \* données) / (# de trames \* (données + entête))

= 12000 / 12032

= 99.7% de données utiles

Mode 4 :

% données utiles = (# de trames \* données) / (# de trames \* (données + entête))

= (300 \* 40) / (300\* (40 + 32))

= 55.6% de données utiles

Le mode 3 est plus performant, il gaspille moins de bande passante en envoyant 99.7% de données utiles.

CEPENDANT, suivant le paquet du mode 3 on peut voir l’erreur Destination unreachable qui est générée par un routeur pour indiquer au host que la destination ne peut etre rejointe.

**Question 3) Discutez de la fiabilité de chaque mode. Selon vous, quel(s) mode(s) est le plus fiable?**

TCP est fiable. Avec TCP il y a confirmation que :

* Le paquet a bien été reçu par le destinataire.
* Si le paquet n’a pas été reçu il est renvoyé.
* Le paquet n’a pas été corrompu.

UDP n’est pas fiable. Avec UDP il n’y a aucune confirmation que les données ont été reçues par le destinataire ou qu’il n’y a pas eu d’erreurs.

En ce sens les modes 1 et 2 sont plus fiables que 3 et 4 car ils utilisent TCP.

**Question 4) Pour les modes secrets utilisant le protocole TCP, vous avez certainement remarqué à la fin de la communication un échange FIN, ACK. Expliquez en quoi consiste cet échange.**

ACK et FIN sont des flags :

**ACK** : de l’anglais « acknowledgement », c’est un accusé de réception que le paquet précédent a été reçu.

**FIN**: envoyé par le host quand il veut mettre un terme a la communication.

Dans le mode 1 on peut voir par exemple :

Étape 1 : le client envoit un paquet au serveur pour signifier qu’il veut mettre fin a la connexion (FIN).

Étape 2 : le serveur lui envoie un accusé de réception (ACK)

Étape 3 : le serveur envoit un paquet au client pour signifier qu’il veut mettre fin a la connexion (FIN).

Étape 4 : le client lui envoie un accusé de réception (ACK)

À ce point-ci, connexion est terminée!