

Département de génie informatique et génie logiciel

INF3405

Réseaux Informatiques

Hiver 2019

Rapport du TP3 : Analyse d’applications client-serveur avec WireShark

Soumis à Émilie Dion-Paquin

Julien Legault 1847125

Abdellah Rahmani 1965352

Le 16 avril 2019

# 7. Analyse de l’application client/serveur du laboratoire 1

**C) Analyse du flot de données de l’application de gestion de fichiers**

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

(Ip.src==192.168.79.143 and ip.dst==192.168.79.145) || (Ip.src==192.168.79.145 and ip.dst==192.168.79.143)

Ce filtre masque toutes les communications n’allant pas du client (192.168.79.143) au serveur (192.168.79.145) ou du serveur au client.

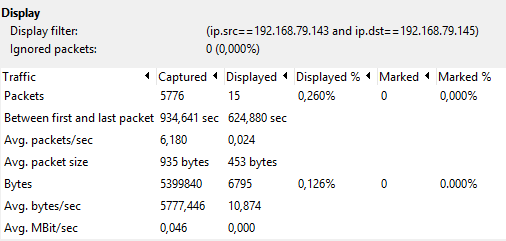
1. **À la lumière de vos observations, dites quel protocole de la couche 4 est utilisé pour la communication entre le client et le serveur.**



Le protocole permettant la communication est le protocole TCP.

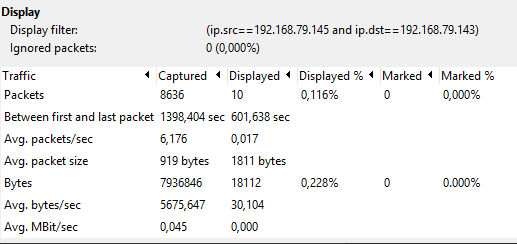
1. **Combien de paquets et d’octets de données ont été envoyés du client vers le serveur et du serveur vers le client?**

Client → Serveur :



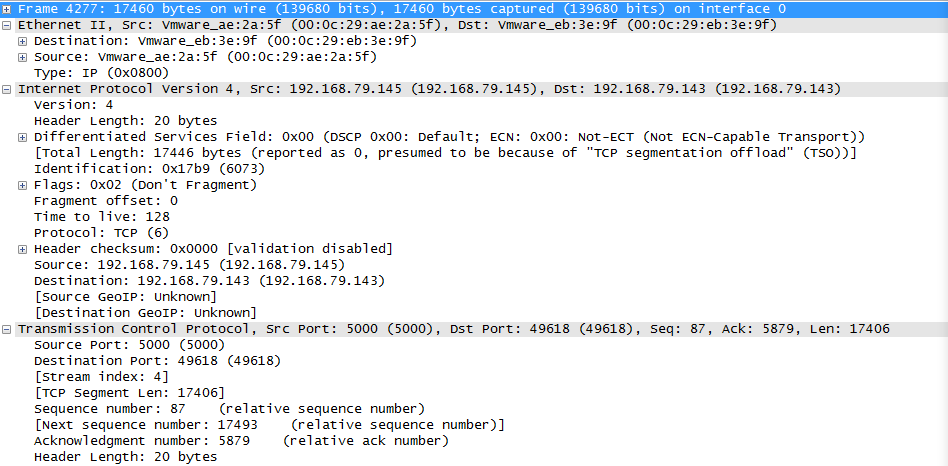
Du client au serveur, 15 paquets ont été envoyés pour un total de 6795 octets.

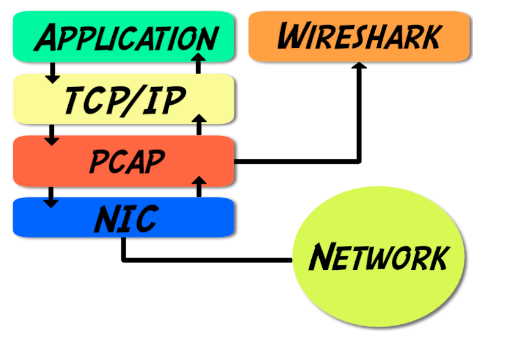
Serveur → Client :



Du serveur au client, 10 paquets ont été envoyés pour un total de 18112 octets.

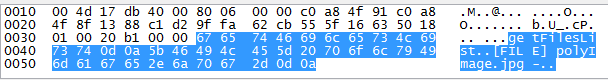
1. **Normalement, le standard IEEE 802.3 limite la taille d’une trame Ethernet à 1518 octets. Dans votre capture Wireshark, existe-t-il des paquets ayant une taille supérieure à 1518 octets? Si oui, expliquez pourquoi et comment ce paquet réussit à transiger sur le réseau alors que sa taille est plus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



Oui, la capture ci-haut montre un paquet faisant plus de 1518 octets. La raison est que le paquet n’a pas été encore fragmenté afin de répondre au standard imposé par la norme IEEE 802.3. La raison pourquoi il n’a pas encore été fragmenté est que wireshark intercepte les paquets entre le TCP/IP et la carte réseau (NIC) qui fragmente les paquets avant de les envoyer sur le réseau. 

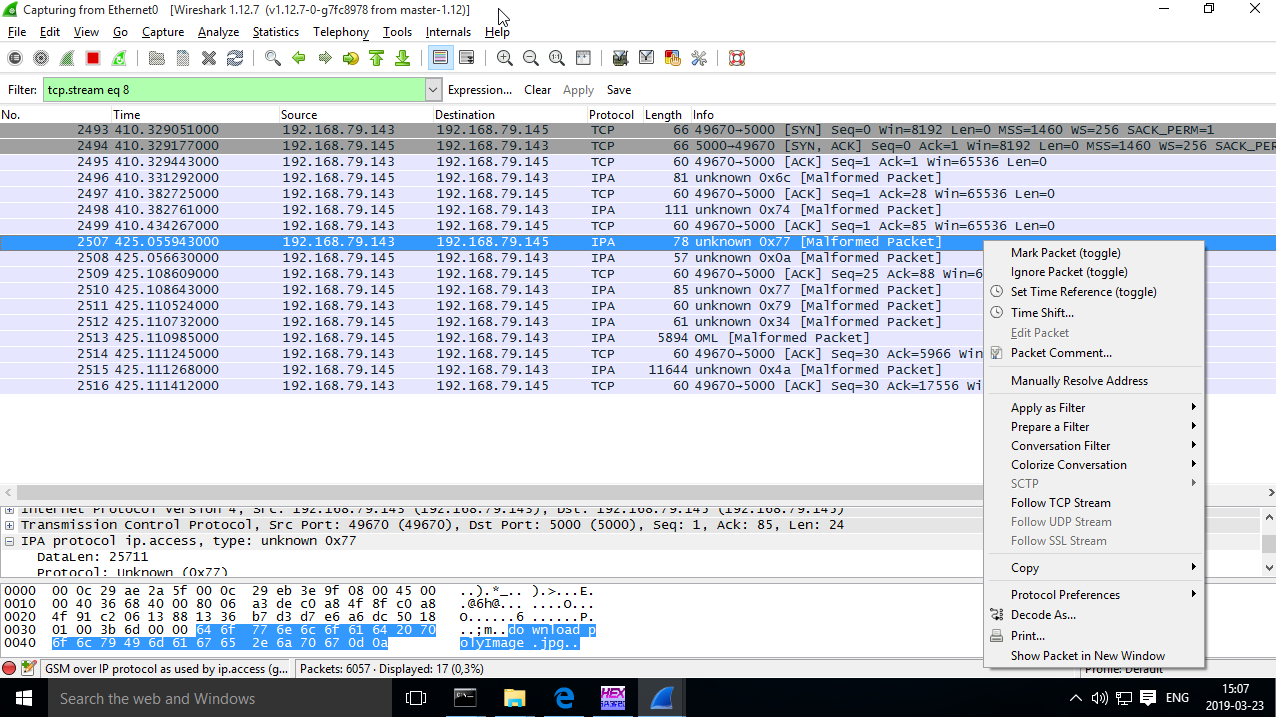
<http://packetbomb.com/how-can-the-packet-size-be-greater-than-the-mtu/>

1. **Quel type d’information êtes-vous capables d’extraire de Wireshark en lien l’exécution de la commande « ls » ? Montrer vos résultats**

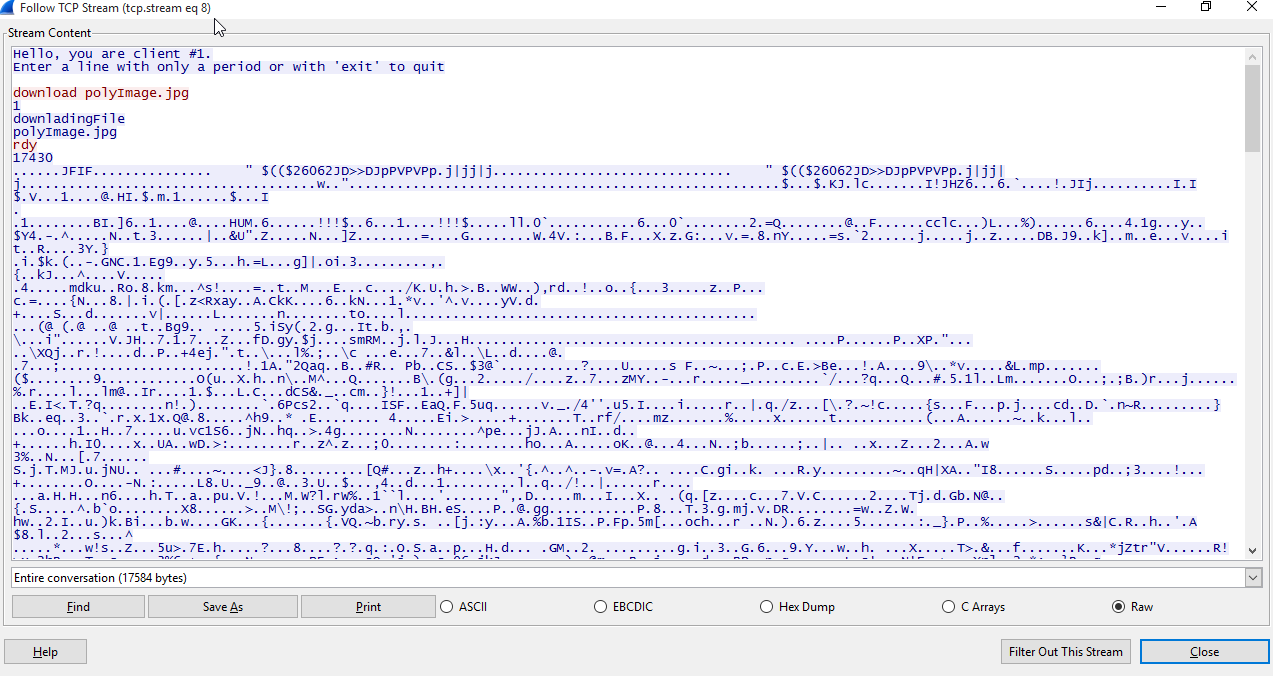


On peut voir que la commande « ls » envoie une valeur encodée sur 37 octets effectuant un « getFiles » du côté serveur retournant ainsi tous les fichiers présents. Dans notre cas, seul l’image « polyImage.jpg ».

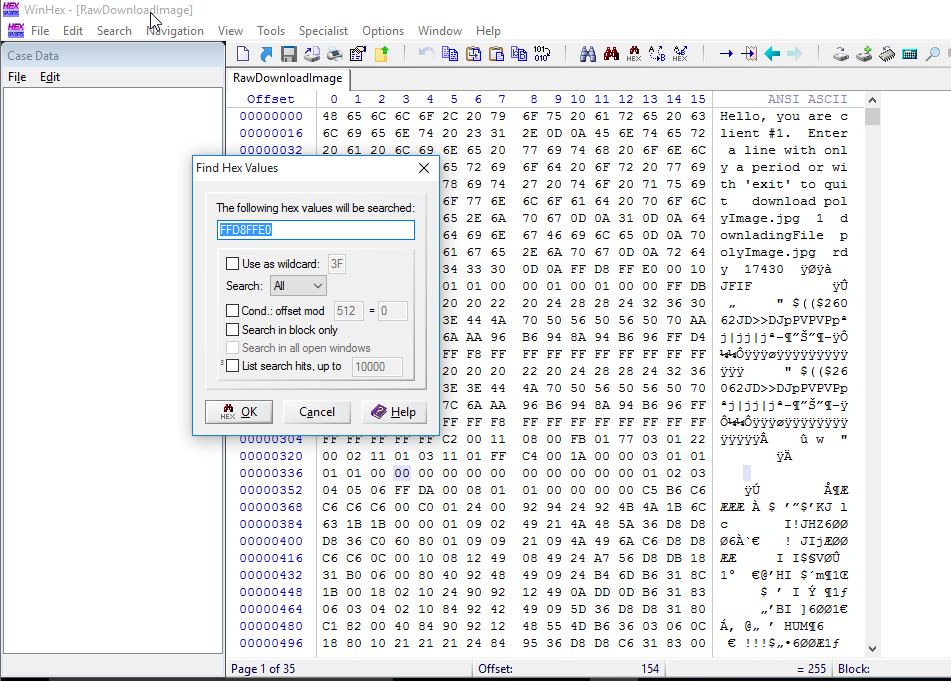
1. **Il est possible, avec Wireshark, d’extraire l’image envoyée par le client ou l’image envoyer par le serveur vers le client. Donnez les étapes à suivre, incluant des captures d’écran montrant chaque étape permettant l’extraction de l’image envoyée du client vers le serveur. Servez-vous des propriétés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



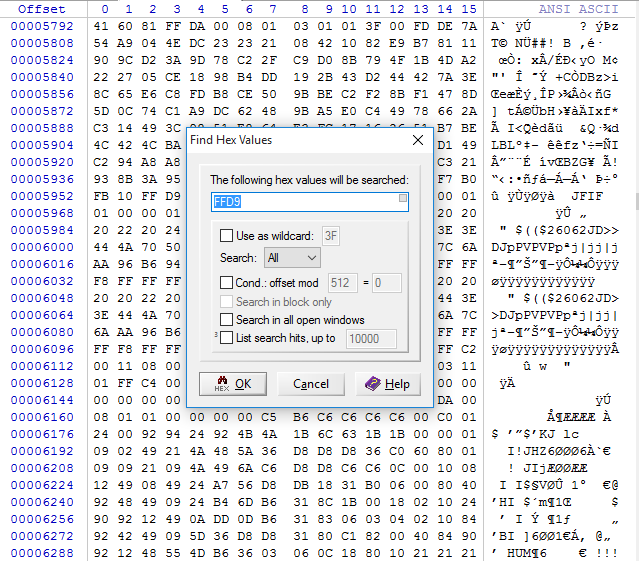
1. Trouvons l’instruction nous intéressant, soit un download PolyImage.jpg et effectuons un Follow TCP Stream pour sauvegarder le paquet.



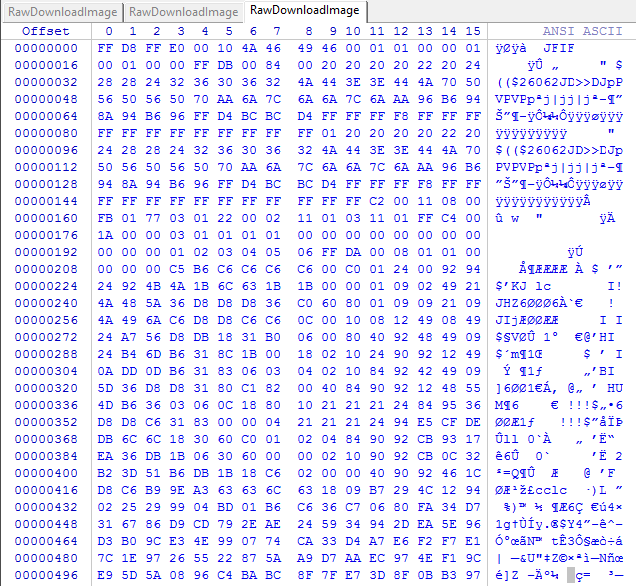
1. Enregistrons le fichier en .Raw pour modifier les données avec le logiciel WinHex.



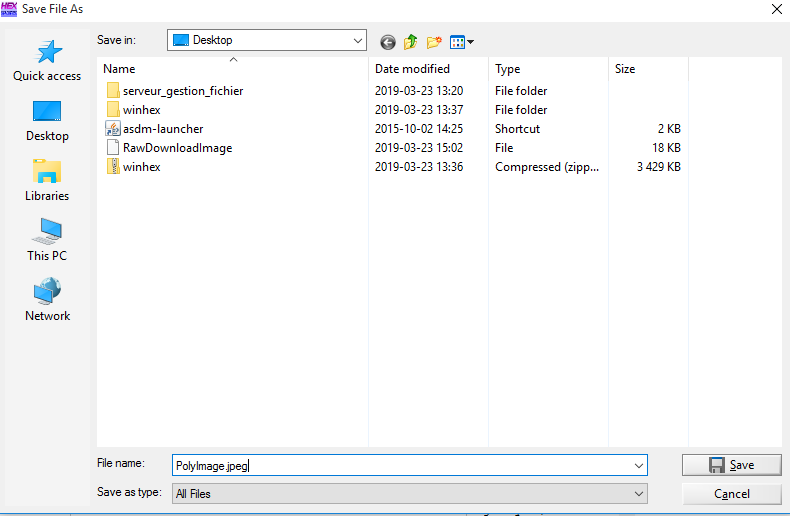
1. Cherche l’entête de début de l’image avec FFD8E0 pour retirer tous les entêtes antécédents identifiant les protocoles de transport utilisés.



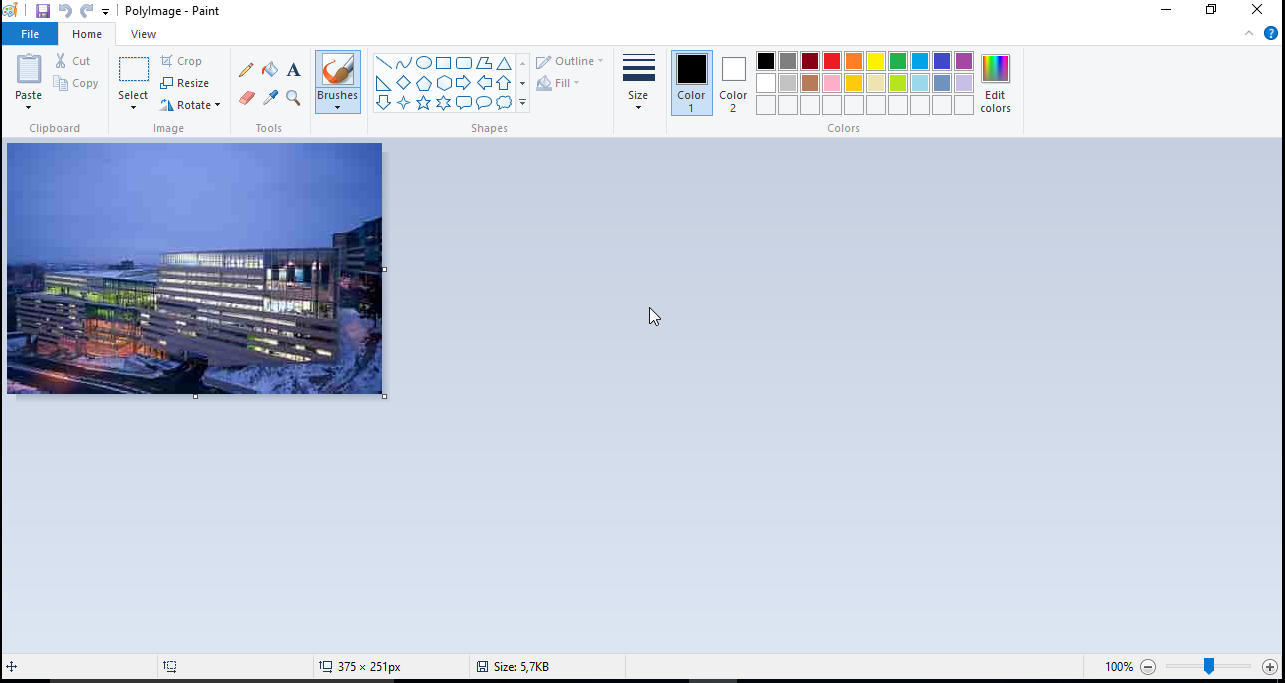
1. Cherche l’entête de fin de l’image avec FFD9 pour retirer toutes les données ne faisant pas partie de l’image.



1. Une fois que nous avons les données ne représentant que l’image, nous pouvons aller choisir l’option Save As.



1. On enregistre les données sous forme jpeg



1. On ouvre l’image avec Paint pour voir que nous avons bien notre image initiale.
2. **Suite à toute cette analyse que pouvez-vous conclure quant à la sécurité de l’application de gestion de fichiers que vous avez développée lors du travail pratique no.1**

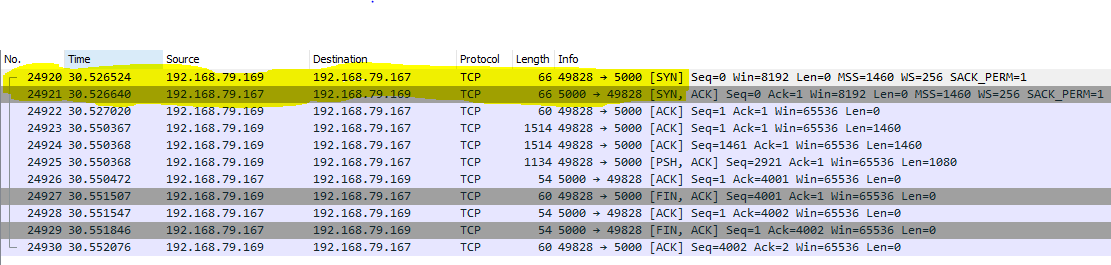
Nous constatons que nos données ne sont pas du tout protégées et que si quelqu’un intercepte l’envoi des données, il sera alors très facile de récupérer son contenu.

# 8. Analyse d’une application client-serveur “secrète”

**B) Mode secret (1, 2, 3 et 4)**

**Mode 1 :**

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 de la couche transport utilisé est le TCP, et le tout premier échange est un échange de type SYN (synchronisation), il permet d’établir la connexion entre le serveur et le client, cet échange est nécessaire dans le cas du TCP, car c’est un protocole avec connexion (**établissement de la connexion** 🡺 transferts de données 🡺 fin de la connexion.) 

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



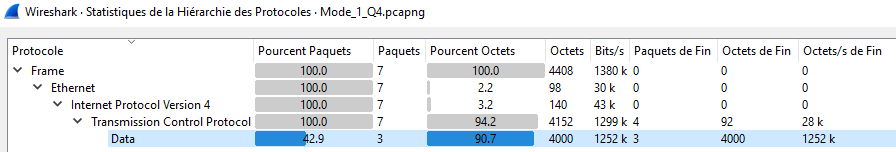
Port TCP source : 49828

Port TCP destination : 5000

1. **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.**

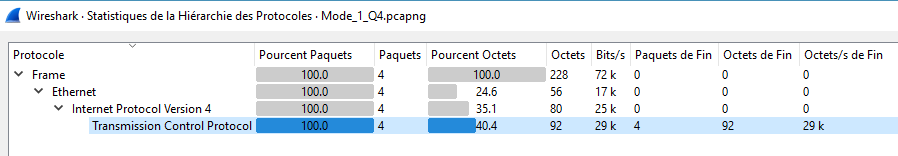
Client→Serveur: (ip.src==192.168.79.169 and ip.dst==192.168.79.167)

3 paquets de données ont été envoyés, l’équivalent de 4000 octets



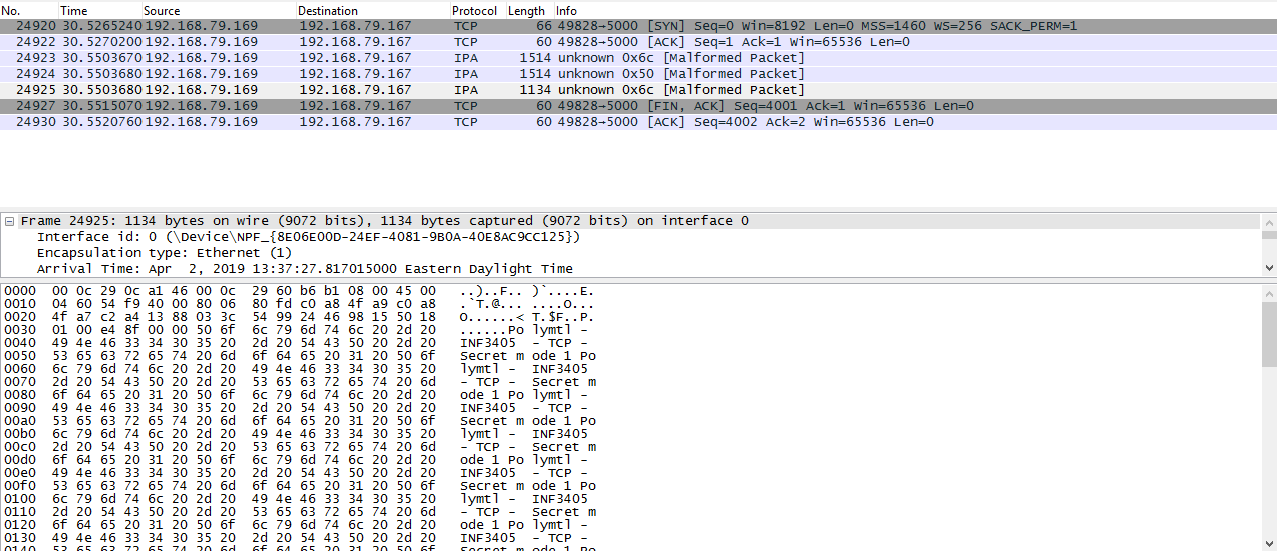
Serveur→Client : (ip.src==192.168.79.167 and ip.dst==192.168.79.169)

100% des paquets sont réservés à l’entête alors il y a 0 paquet contenant des données, et donc 0 octet.



1. **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?**

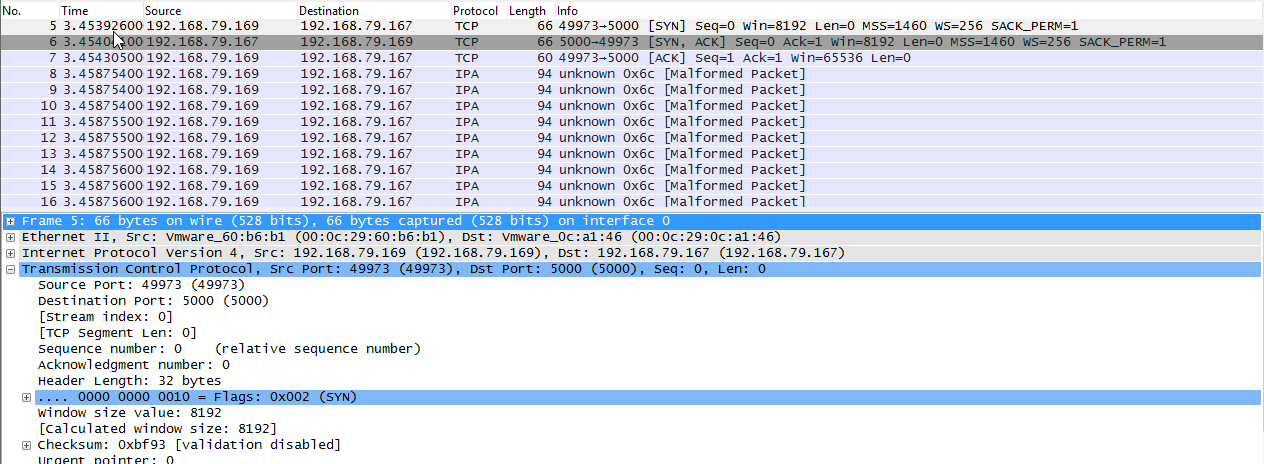
Le client envoie le message secret « Polymtl – INF3405 – TCP – Secret – mode 1 … » en un seul bloc de donnée, une fragmentation automatique s’est effectuée, 3 itérations sont donc faites.



**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 de la couche transport utilisé est le TCP, et le tout premier échange est un échange de type SYN (synchronisation), il permet d’établir la connexion entre le serveur et le client, cet échange est nécessaire dans le cas du TCP, car c’est un protocole avec connexion (**établissement de la connexion** 🡺 transferts de données 🡺 fin de la connexion.)



1. **En vous basant sur les informations recueillies par Wireshark, indiquez le port source et destination utilisé par la couche 4.**



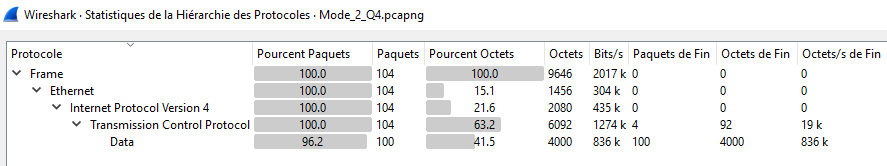
Port TCP source : 49973

Port TCP destination : 5000

1. **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.**

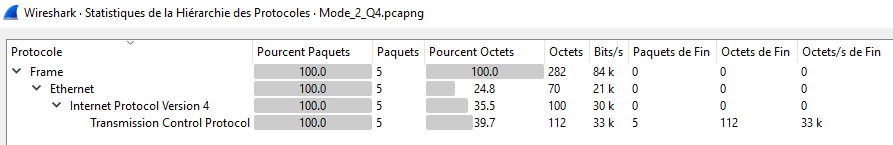
Client →Serveur: (ip.src==192.168.79.169 and ip.dst==192.168.79.167)

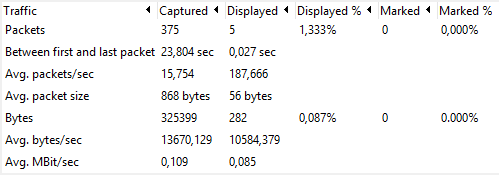
100 paquets de données ont été envoyés, l’équivalent de 4000 octets



Serveur→Client: (ip.src==192.168.79.169 and ip.dst==192.168.79.167)

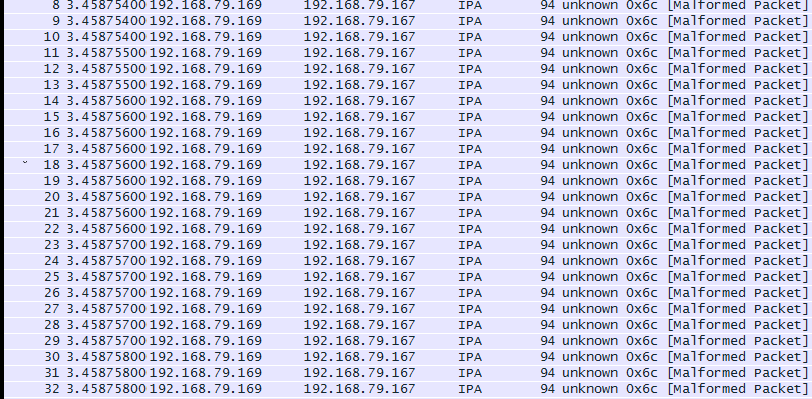
100% des paquets sont réservés à l’entête alors il y a 0 paquet contenant des données, et donc 0 octet.





**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?**

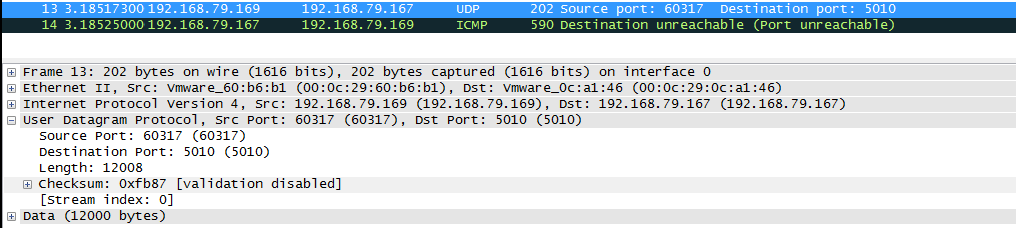
Le client envoie la même quantité de données qu’en mode 1, mais partagées dans 100 paquets au lieu de 3, 100 itérations sont faites.



**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?**

C’est le protocole UDP qui est utilisé. Dans le cas d’UDP, cet échange n’aura pas lieu, car UDP est un protocole qui n’utilise aucune synchronisation.



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



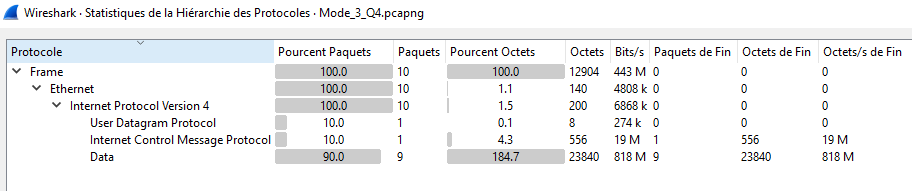
Port UDP source : 60317

Port UDP destination : 5010

1. **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.**

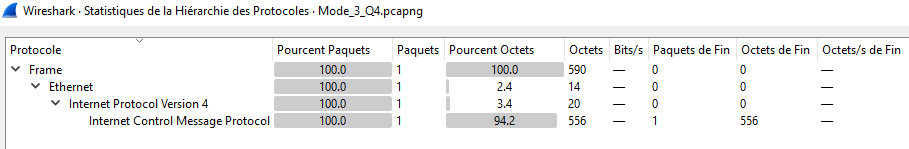
Client →Serveur : (ip.src==192.168.79.169 and ip.dst==192.168.79.167)

9 paquets de données envoyés, l’équivalent de 23840 octets

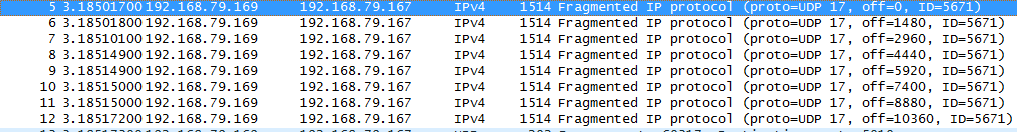


Serveur→Client: (ip.src==192.168.79.167 and ip.dst==192.168.79.169)

Aucun paquet ni octet transmis pour les mêmes raisons.



1. **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?**

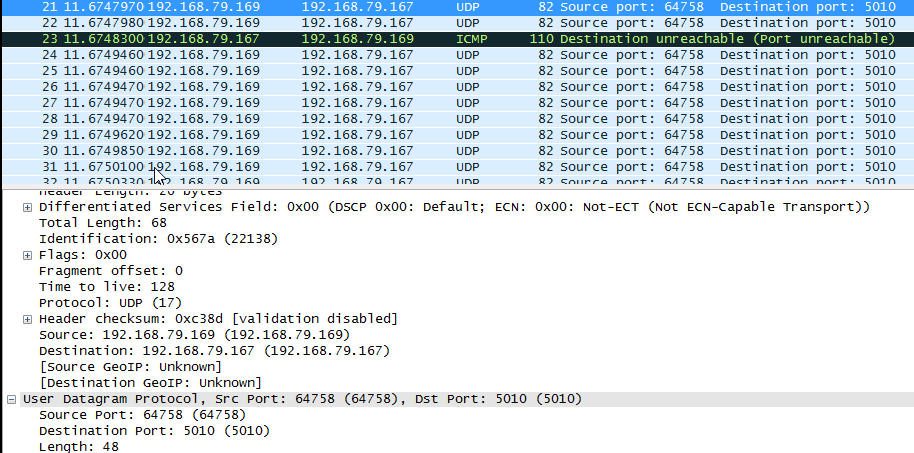


Le client envoie toutes les données au serveur en un seul bloc, et comme la taille d’un paquet dépassera la taille maximale dans ce cas, une fragmentation automatique se fait, par conséquent, les données sont envoyées en 9 itérations.

**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 a lieu? Pourquoi?**

C’est le protocole UDP qui est utilisé. Dans le cas d’UDP, cet échange n’aura pas lieu, car UDP est un protocole qui n'utilise aucune synchronisation.



1. **En vous basant sur les informations recueillies par Wireshark, indiquez les ports sources et destination utilisés par la couche 4. (0.5 point)**



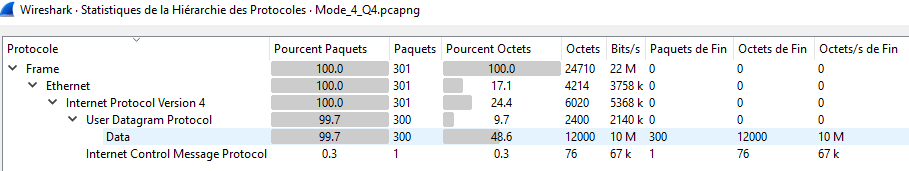
Port UDP source : 64758

Port UDP destination : 5010

1. **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.**

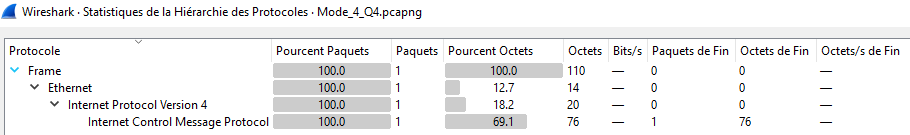
Client→Serveur : (ip.src==192.168.79.169 and ip.dst==192.168.79.167)

300 paquets de données envoyés, l’équivalent de 12000 octets



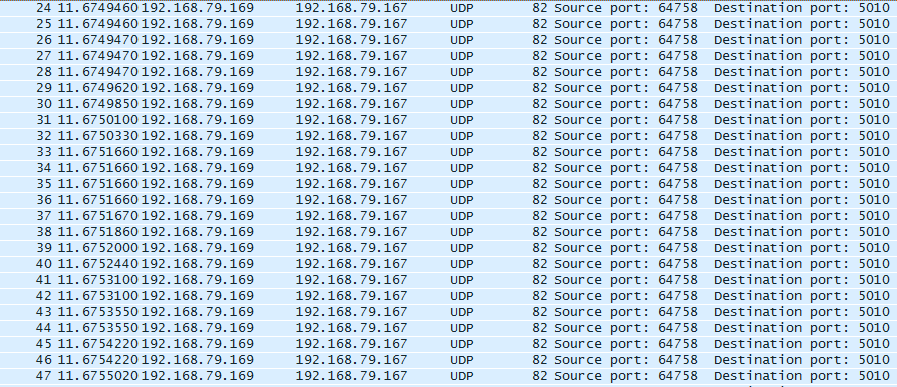
Serveur→Client : (ip.src==192.168.79.167 and ip.dst==192.168.79.169)

Aucun paquet ni octet transmis contenant des données. Il n’y a uniquement que les entêtes



**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?**

Contrairement au mode 3, le client divise les données en plusieurs blocs envoyées en 300 itérations



**C) Analyse des performances et protocole TCP**

**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?**

Dans le mode 1, on envoie les données en utilisant peu de paquets (3 paquets pour 4000 octets), tandis qu’en mode 2, 100 paquets sont utilisés pour envoyer la même quantité de données, et vu que les deux modes utilisent le même protocole : TCP, le mode le plus performant serait le mode 1, car la quantité des données inutiles (les entêtes) envoyées sur le réseau est faibles comparée au deuxième mode.

1. **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?**

En mode 3, l’envoi des données se fait d’un seul coup, tandis qu’en mode 4, les données sont envoyées en petits blocs, pour comparer les deux modes en termes de performance, le 3e mode serait meilleur, car il consomme moins de bande passante (moins d’entêtes envoyés sur le réseau)

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

Les modes 1 et 2 utilisent le protocole TCP qui utilise le mode connecté, il garantit donc le transport des données vers leur destination, en relançant l’envoie à nouveau en cas de problèmes de transmission (pertes, paquets défectueux …), par contre, les modes 3 et 4 utilisent le protocole UDP qui est plus rapide, mais avec le risque de perte de données (pas d’acquittement ), nous pouvons donc dire qu’en termes de fiabilité, le mode 2 serait le meilleur, car en manipulant des blocs de petites tailles, la détection et la correction des problèmes de transmission seraient plus rapides, ensuite vient le 1er mode, et puis le 4e mode, car en utilisant des petits blocs de donnés avec UDP, les pertes seraient minimes , et en dernier vient le 3e mode.

1. **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.**

Cet échange consiste en la phase de terminaison d'une connexion : en fin de transmission des données, la fin de la connexion nécessite un échange FIN et ACK dans les deux sens client 🡨🡪 serveur, c’est le principe du handshaking en quatre temps entre le client et le serveur.