Rapport du PROJET de RSA

-

Programmation d'un serveur proxy

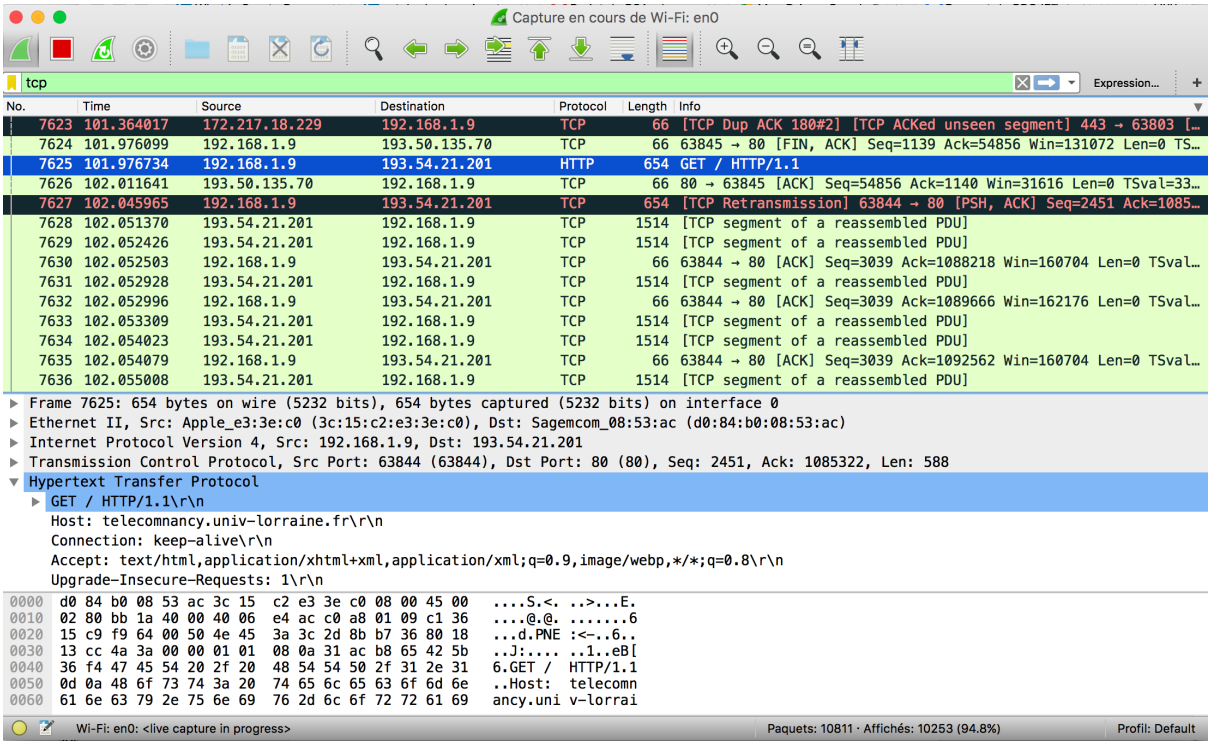
Rédigé par
Jean-Baptiste DOMINGUEZ, Henry

Table des matières

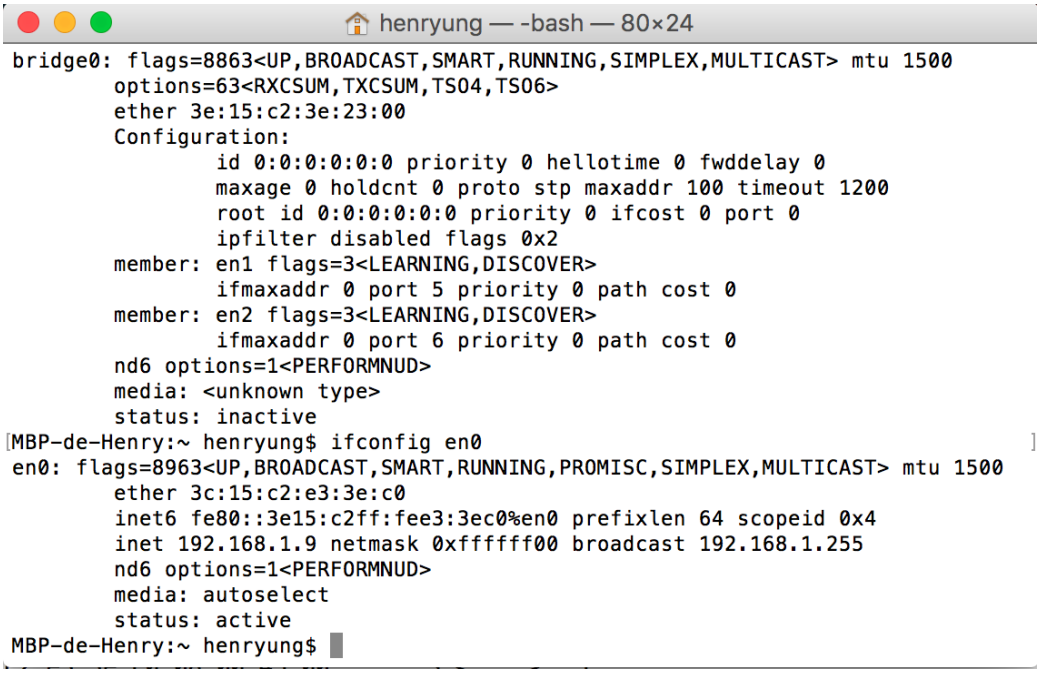
Question 1 ..... **3**
Question 2 ..... **4**
Question 3 ..... **5**
Question 4 ..... **5**
Question 5 ..... **6**
Question 6 ..... **6**

Question 1

A l'aide du logiciel Wireshark nous avons analysé les échanges entre notre machine et un serveur web. Voici la capture d'écran que nous obtenons.



C'est le paquet nº7625 qui nous intéresse. Nous observons que l'adresse IP de destination du serveur web à l'adresse http.host == "telecomnancy.univ-lorraine.fr" est 193.54.21.201 et que notre adresse IP est 192.168.1.9 Pour en être sûr nous utilisons la commande ifconfig dans le terminal

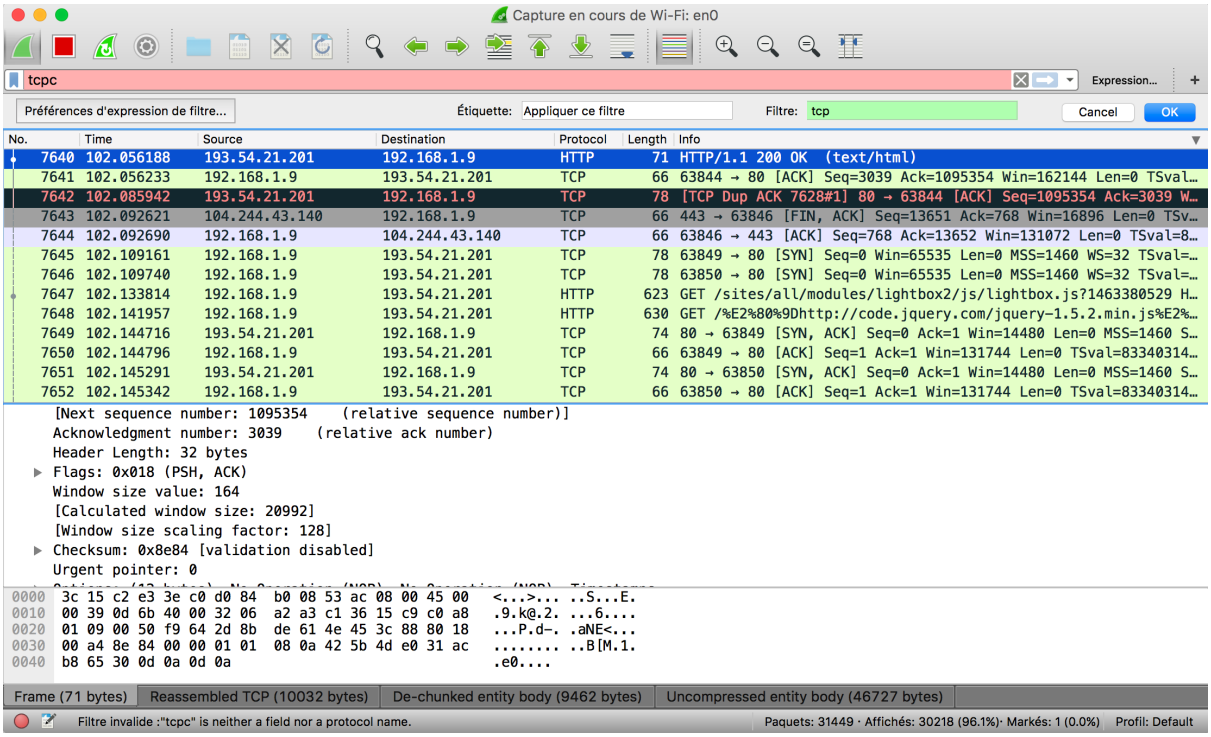


A la ligne inet on observe que notre adresse IP est bien 192.168.1.9 nous utilisons le netmask 0xffffffff00 et notre adresse de broadcast est 192.168.1.255

Afin de mieux voir ce qui nous intéresse nous utilisons le filtre “ip.addr == 193.54.21.201” qui nous permet de n’afficher que la “discussion” entre notre client et le serveur web de telecomnancy.eu

Pour accéder au site on fait une requête HTTP d’une longueur de 654 bits :”GET / HTTP/1.1” on utilise l’Internet Protocol Version 4 (IPv4) et la Transmission Control Protocol (TCP) afin d'établir une connexion et garantir la bonne transmission des données.

On effectue une demande de connexion ainsi le serveur web et le client s’échange des données lorsque la requête GET est satisfaite on reçoit :”HTTP/1.1 200 OK (text/html)”.



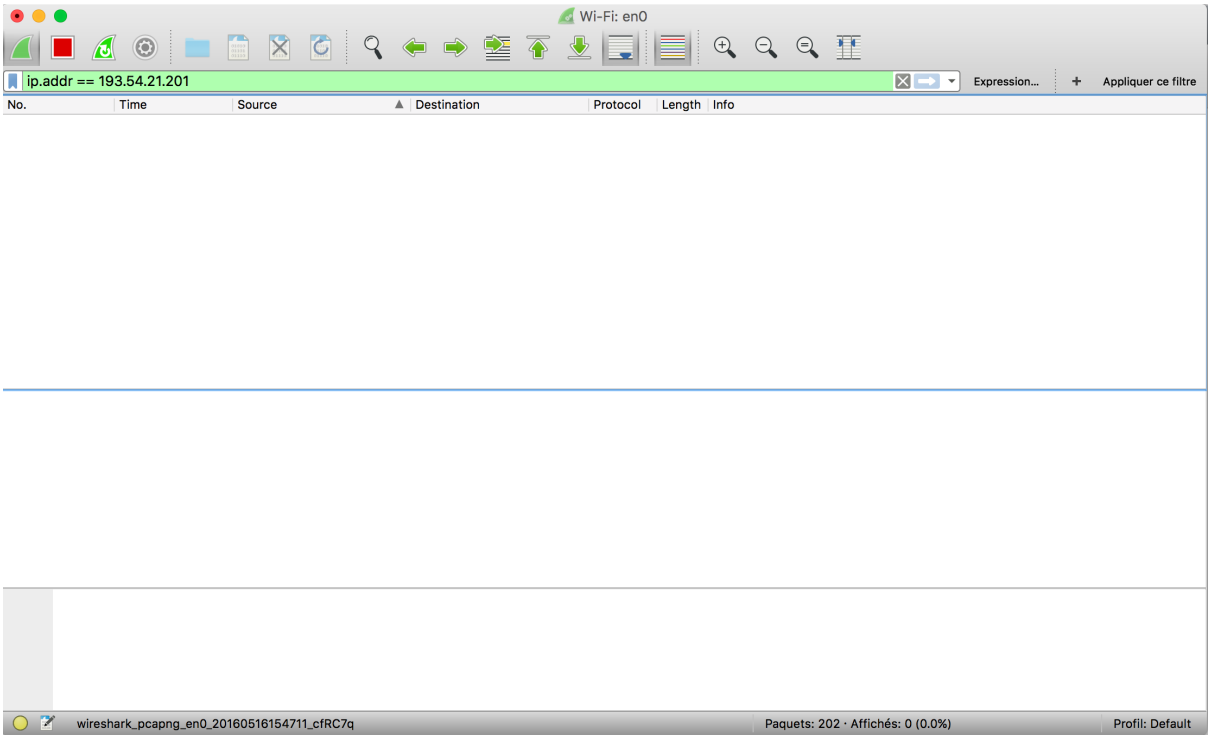
On cherche ensuite le three-way handshake qui caractérise TCP et qui garantie que la connexion a bien été effectué. On trouve les réponses respectives des paquets [SYN] numérotés 7645 et 7646 au paquet nº7649 [SYN, ACK] et nº7650 [ACK] ainsi qu’au paquet nº 7651 [SYN, ACK] et nº 7652 [ACK]. Avant on observe que deux requêtes HTTP “GET”ont été effectuées pour récupérer des données. Globalement, plus d’un milliers de paquets on été échangés entre le client et le serveur web. Le dernier paquet portant le numéro 8181.

Question 2

Pour configurer notre machine afin d’utiliser le serveur proxy que nous avons programmé nous allons dans les paramètres réseaux. Nous entrons l’adresse et le numéro de port adéquat. Soit 127.0.0.1 pour l’adresse IP et 1330 port le numéro de port.

Question 3

Comme le serveur proxy n'est pas configuré nous nous attendons à voir aucun échange entre notre machine et le serveur web. Ainsi l'essai de connexion entre le client et le serveur est voué à échouer. En effet une capture d'écran confirme qu'aucune donnée n'est échangée.



Nous observons alors une page blanche. Cela paraît banal mais montre qu'une "simple" configuration de proxy HTTP permet de contrôler la connexion aux différents sites web présent sur Internet. Nous allons maintenant concevoir et implanter un serveur proxy afin de gérer d'abord un client et ensuite plusieurs client.

Question 4

La communication entre un client et un serveur en passant par un proxy est représentable de la manière suivante.

Client >>>>> Proxy >>>> Serveur web >>>>> Proxy >>>>>> Client

L'algorithme général du proxy est le suivant : on attend qu'un client fasse une requête HTTP. Lorsque le serveur proxy reçoit une requête il envoie cette requête au serveur web. Ensuite, il attend de recevoir la réponse du serveur web qu'il transmet par la suite au client. On répète cette procédure jusqu'à ce que le client n'ait plus de requêtes et met un terme à la communication.

Question 5

Nous avons implémenté un proxy qui permet de gérer les requêtes d'un client. Puis nous avons amélioré notre programme afin qu'il gère simultanément plusieurs clients.

Question 6

Pour la gestion simultanée de plusieurs clients nous avons utilisé plusieurs threads. Nous avons fait ce choix en nous interrogeant sur plusieurs critères. Nous connaissons en tout trois manières de gérer des requêtes simultanées. La première que nous avons étudié en TD consiste à créer une liste et de servir les clients au fur et à mesure avec les "raw sockets". Cela ne respecte pas ce qui est demandé donc nous avons écarté cette possibilité. En cours, nous avons étudié les processus dupliqués grâce à l'instruction fork(). L'utilisation de cette instruction présente un avantage principal, il est facile à implanter par rapport à la troisième alternative que nous connaissons : le multi-threading. Malheureusement, l'utilisation de l'instruction fork() est gourmande en ressource car elle duplique intégralement le programme. C'est donc par souci d'économie de ressources que nous avons choisi d'implanter un serveur proxy qui gère plusieurs clients à la fois en utilisant le multi-threading.