La couche Transport

14 novembre 2024

1 Protocole UDP

Dans la couche inférieure, on route les datagrammes IP d'un routeur à un autre, mais aucune distinction n'est faite entre les services qui émettent et reçoivent. On rajoute donc une nouvelle couche de données qui contient cette information supplémentaire : le **port**.

Codé en tant qu'entier positif sur 16 bits, le port permet d'identifier de manière unique un processus qui va communiquer vers l'extérieur. C'est le SE qui octroie l'accès à un port à une application, et qui doit gérer la file d'attente des paquets ; jusqu'à ce qu'une application les lise. Les numéros 1 à 1023 sont réservés à des services connus. Il est impossible d'implanter de manière durable un serveur dont le port est compris dans cet intervalle.

Il n'y a aucun mécanisme supplémentaire, comme le montre l'en-tête dans la figure 1, il y a que les numéros de port de l'application source, et de la destination, la longueur de message, et un checksum (facultatif) pour assurer l'intégrité des données à l'intérieur.

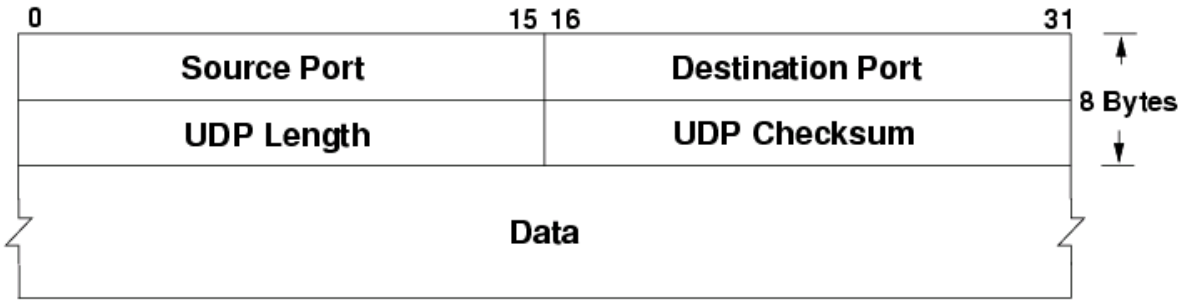

Figure 1 – En-tête UDP

2 Protocole TCP

TCP est un protocole de communication fiable à flot d'octets orienté connexion.

La connexion entre un client et un serveur s'effectue avec une ouverture en trois temps (*three-way handshake*). Le point de connexion d'un serveur, nommé *socket*, attend alors une demande de connexion d'un client.

1. Le client, souhaitant se connecter, envoie un segment SYN au serveur qui contient le numéro de séquence à utiliser comme numéro d'initialisation.
2. Le serveur retourne message contenant le message SYN du client, et un accusé de réception ACK.

3. Le client commence à envoyer des données, jusqu'à terminaison de la liaison, qui s'effectue avec un *three-way handshake* contenant le drapeau FIN qui signifie que l'émetteur n'a plus de données à transmettre.

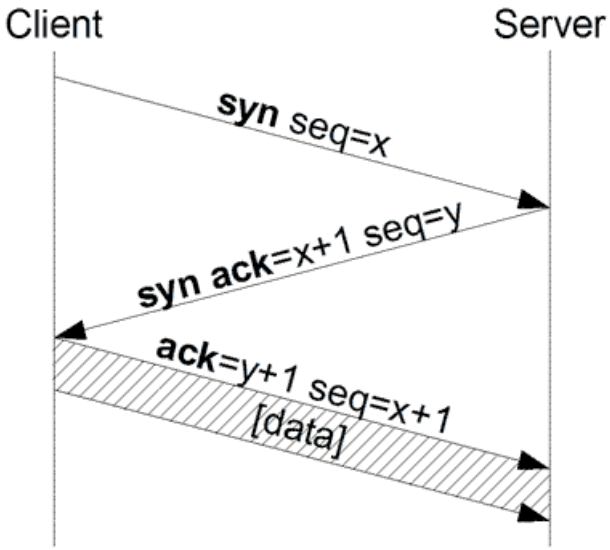

FIGURE 2 – Schéma temporel du Three-way handshake
Source: https ://commons.wikimedia.org/wiki/File :Tcp-handshake.png

2.1 Mécanisme d'acquittement

Le premier paquet envoyé par le client est marqué par un nombre de séquence (SEQ) aléatoire x, qu'on ramènera à 1 pour faciliter la compréhension. Puis ce nombre augmente du nombre d'octets qui ont été émis : si le client émet n octets, son prochain paquet aura pour numéro SEQ $x+n$. Du côté du serveur, celui-ci accuse réception d'un certain nombre d'octets. En recevant les n octets et la valeur initiale x, le serveur retourne la valeur $x+n$ dans le paramètre ACK, indiquant que les n octets ont été reçus.

Ce mécanisme est cumulatif : la valeur de ACK détermine le numéro SEQ attendu pour le prochain paquet. Si on reçoit un segment avec un numéro SEQ plus grand que celui attendu, on le conserve, mais sans l'acquitter, on attend d'avoir reçu tous les segments pour le faire. Si un segment n'est pas acquitté, il est considéré comme perdu et il doit être retransmis (la transmission reprend au dernier octet acquitté, donc les éventuels segments qui ont quand même été reçus seront détruits).

2.2 Contrôle du flux

Chaque machine communique la taille de sa mémoire tampon à son interlocuteur, qu'on appelle fenêtre. Elle correspond à la quantité d'information qu'on peut envoyer sans attendre l'acquittement. Si l'interlocuteur a une fenêtre de 32000 octets, alors il est possible d'émettre jusqu'à 32000 octets sans se préoccuper de la réception.

2.3 Retransmission

On rappelle que le *round time trip*, qu'on notera RTT par la suite, est le temps entre l'émission d'un segment et son acquittement.

FIGURE 3 – Illustration du principe de la fenêtre TCP

Le temps de temporisation avant retransmission doit être choisi avec soin :

— s'il est trop court, on risque d'envoyer beaucoup de paquets inutiles ;

— s'il est trop long, alors la durée de transmission risque d'augmenter fortement.

L'algorithme de Jacobson permet de modifier dynamiquement ce temps. On évalue le RTT à l'aide de timestamps, et on le pondère par un coefficient a compris entre 0 et 1, ainsi que l'ancien RTT estimé pour le calcul :

$$RTT_{n+1} = a \times RTT_n + (1 - a)RTT_{nv} \tag{1}$$

En général, on prend a proche de 0,9.

On calcule ensuite la temporisation à partir d'un coefficient $b > 1$, mais qui doit rester assez proche de la valeur réelle.

$$T = b \times RTT_{n+1} \tag{2}$$

L'idée est de rendre b variable : à chaque fois qu'un nouveau ACK est reçu, on calcule l'écart du nouveau RTT avec le RTT estimé (Δt), pondéré par a :

$$D = (1 - a) \times \Delta t \tag{3}$$

La plupart des machines exploitent cette valeur de T :

$$T = RTT \times 4D \tag{4}$$

2.4 Détail de la trame

On détaillera ci-dessous les éléments d'une trame TCP (voir figure 4). Son en-tête contient au minimum 20 octets, mais peut augmenter avec des éléments optionnels.

— Les champs Port source et Port destination (16 bits chacun) représentent respectivement le numéro d'application de l'expéditeur et le numéro d'application du destinataire.

— Le champ Numéro de séquence (32 bits) contient le nombre d'octets envoyés par la machine émettrice.

— Le champ Numéro d'acquittement (32 bits) contient le nombre d'octets qui ont été reçus de la part de l'autre machine.

— Le champ Offset (4 bits) indique le numéro du mot de la trame à partir duquel les données de l'application débutent. Ce champ contient très souvent la valeur 5 car un en-tête sans option fait 20 octets.

— Le champ Réservé (6 bits) est, comme son nom l'indique, présent dans le cas où on souhaite rajouter un ou plusieurs champs. Ses 3 premiers bits sont devenus (RFC 3168 / RFC 3540) un champ ECN ou NS, signalant la présence de congestion

— Le champ de Contrôle (6 bits), contenant les données suivantes :
  — URG : un pointeur de données urgentes ;
  — ACK : indique que le paquet est un accusé de réception ;
  — PSH : indique que les données sont à envoyer tout de suite ;
  — RST : indique une rupture anormale de la connexion ;
  — SYN : indique une demande de synchronisation ;
  — FIN : indique une demande de fin de connexion.

— Le champ Fenêtre (16 bits) indique le nombre d'octets disponibles dans la mémoire tampon de réception ;

— Le champ Total de contrôle (*Checksum*) (16 bits) calculé sur l'ensemble de l'en-tête ;

— Le champ Position d'urgence (16 bits) qui indique le nombre d'octets, en partant du premier, qui sont considérées comme des données à transmettre de manière urgente (en premier)

Nous laisserons de côté le champ Options et le remplissage.

La figure 6 présente une trame TCP et plus particulièrement la partie *flags*, où celui de la synchronisation est à 1. On prendra soin de ne pas envoyer de données compromettantes claires, comme décrit dans la figure 7.

La fiabilité vient de l'utilisation du PAR (*Positive Acknowlegment with Retransmission*) qui consiste à retransmettre un **segment** dont on n'a pas reçu l'accusé de réception après un certain temps.

Exercice 1

On réalisera un rapport écrit avec Word, ou un éditeur LaTeX, exporté en PDF.

1. Ouvrir un navigateur (Firefox ou Google Chrome de préférence), puis lancer leur inspecteur.
2. Ouvrir Wireshark et lancer une écoute.
3. Ouvrir https://random.dog/woof.json, puis avec l'adresse obtenue, lancer la requête pour obtenir une photo d'un chien aléatoire. Lorsque la photo est chargée, mettre fin à l'écoute et enregistrer le fichier Wireshark.
4. Quels sont les protocoles mis en jeu pour obtenir la page ?
5. Indiquer le protocole dont le paquet contient l'adresse IP du serveur. Retracer l'échange de votre requête jusqu'à la réponse.
6. Faire une capture d'écran des octets (donnés en hexadécimal) du premier segment TCP émis. Mettre en évidence sur l'image les données des autres protocoles encapsulés à l'intérieur. En s'aidant de la fenêtre d'étude des protocoles de Wireshark, donner leur nom, vérifier et indiquer votre adresse IP ainsi que le port utilisé pour la connexion.
7. Indiquer le nombre total de segments de la transaction.

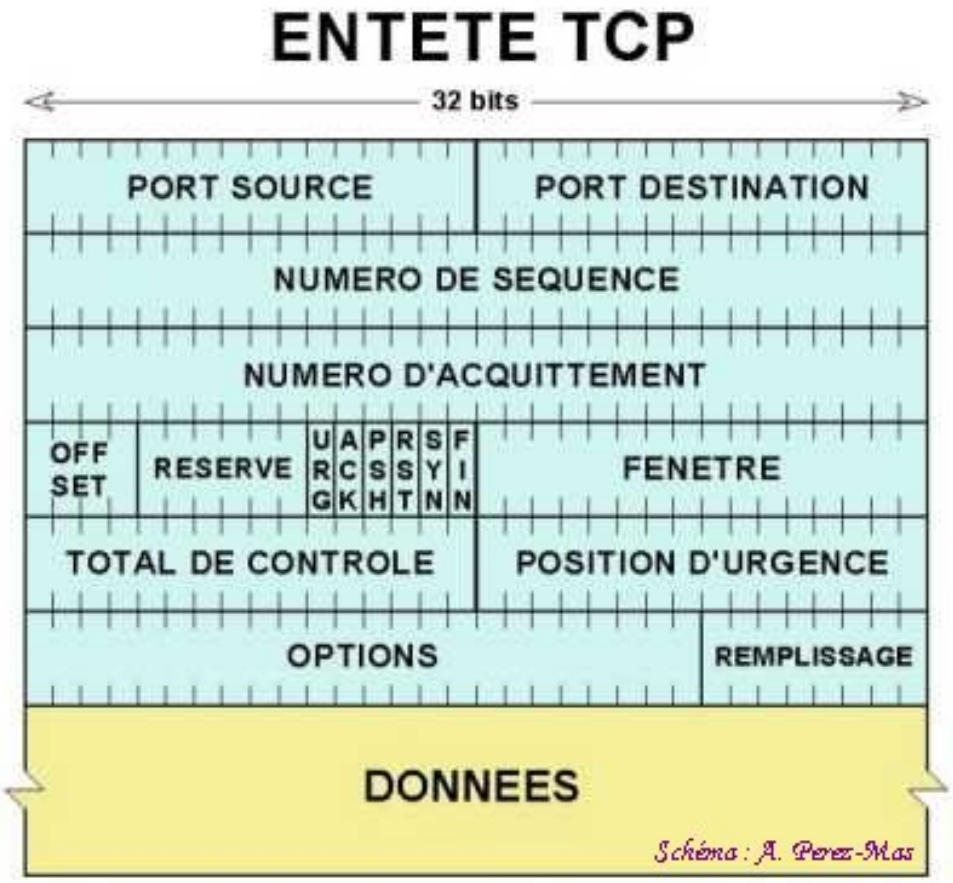

FIGURE 4 – Détail de l'en-tête TCP
Source: http ://arsene.perez-mas.pagesperso-orange.fr/reseaux/tcpip/tcp.htm

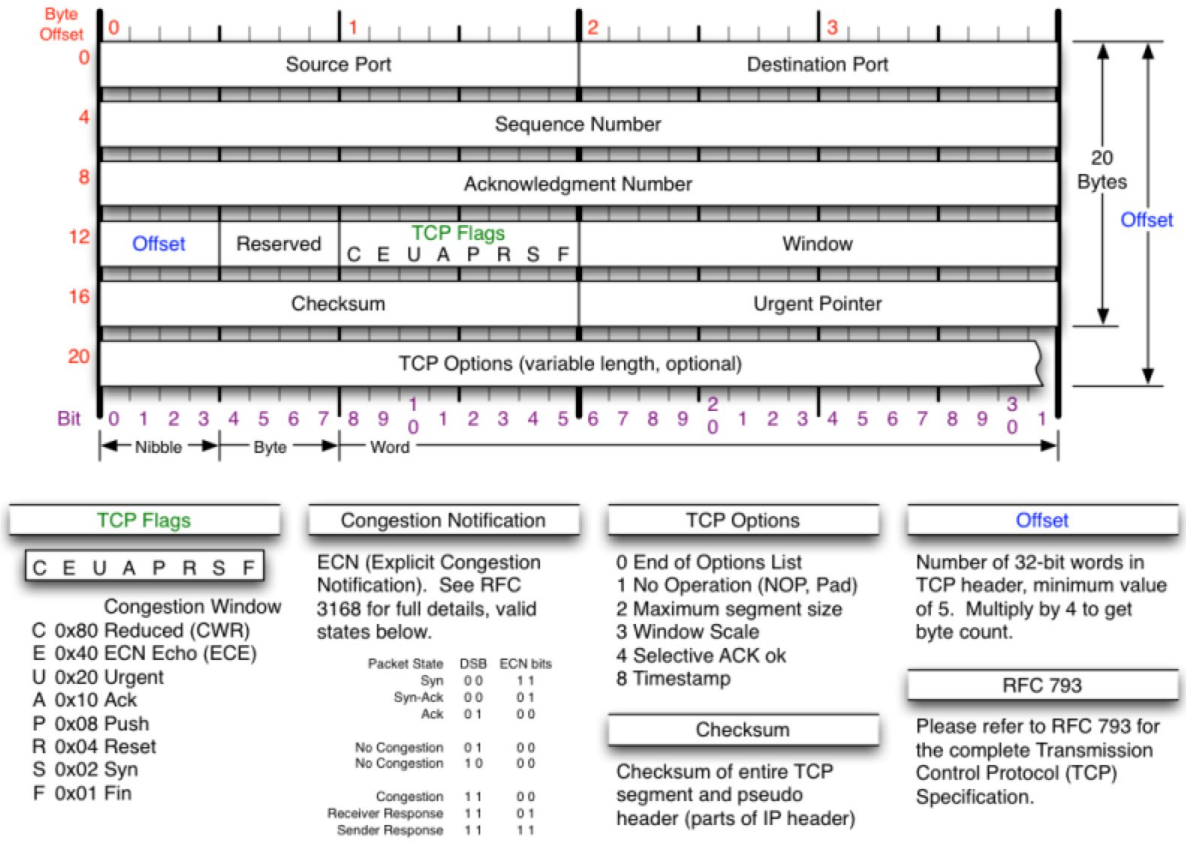

FIGURE 5 – Détail de l'en-tête TCP revisité : 3 nouveaux drapeaux
Source: https ://nmap.org/book/tcpip-ref.html

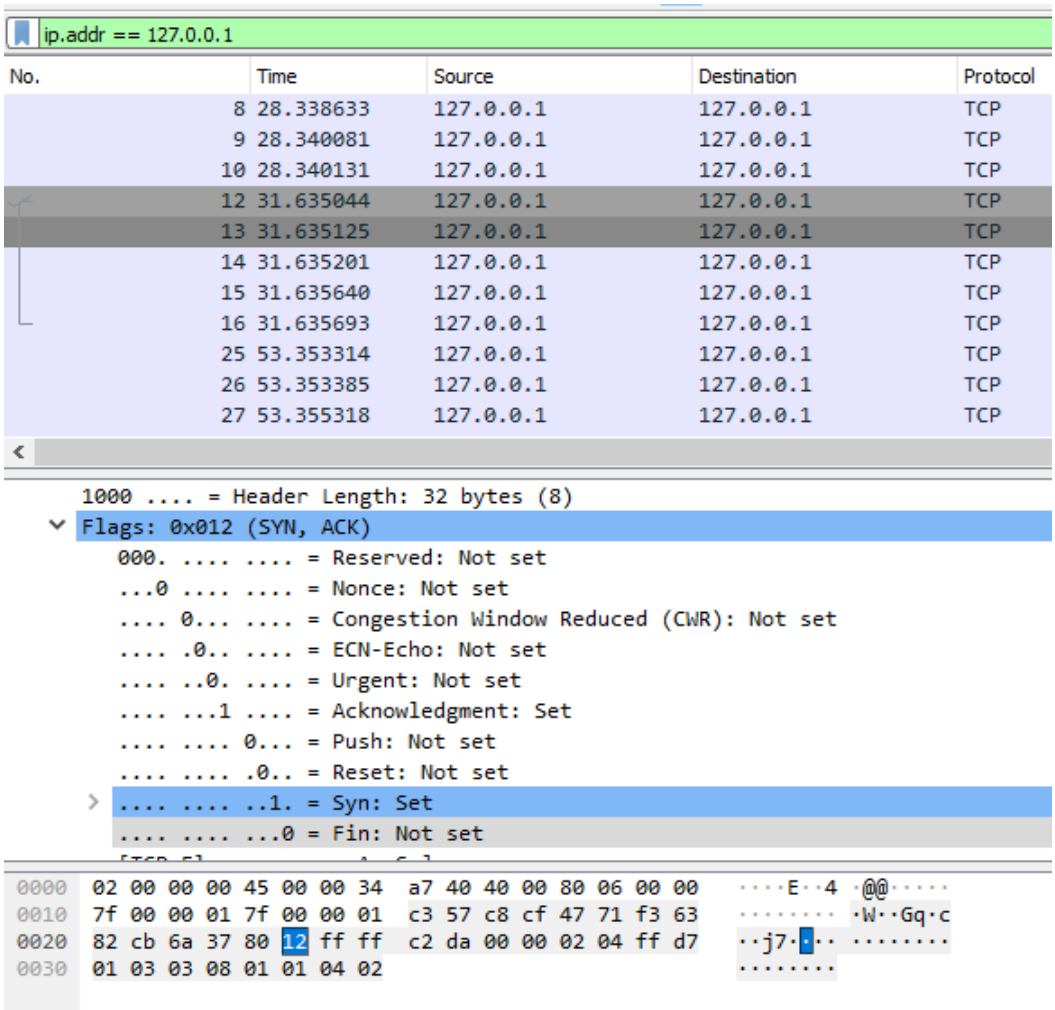

FIGURE 6 – Capture de la séquence d'initialisation d'une connexion client-serveur Python avec Wireshark

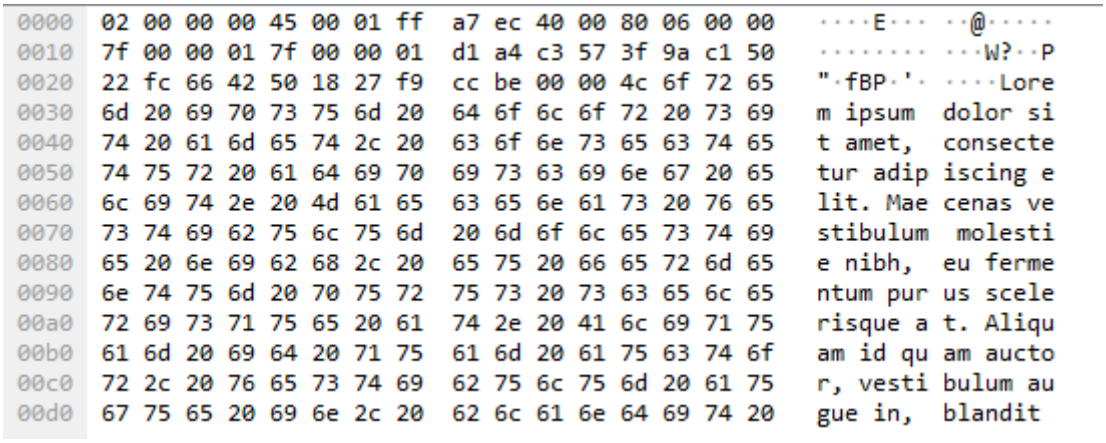

FIGURE 7 – Paragraphe de Lorem Ipsum intercepté et parfaitement lisible.