utbm

Etude des protocoles de transport TCP & UDP

BOUCARD Damien
BOURGON Jean-Xavier
LOURD Rodolphe

SOMMAIRE

Introduction

Le but de ce TP est d'étudier le fonctionnement de la couche transport du protocole TCP/IP. Pour cela, nous allons étudier le fonctionnement des deux modes de connexion : TCP (connecté) et UDP (datagramme). Nous observerons aussi la réaction du protocole face à un afflux d'information, des coupures réseau, etc.

TCP (*Transmission Control Protocol*) est un protocole orienté connexion. Il opère donc avec un contrôle de transmission des données pendant une communication établie entre deux postes. Les données sont donc envoyées sous forme de flot en sachant que la machine réceptrice envoie des accusés de réception lors de la communication et que la machine émettrice est garante de la validité des données qu'elle envoie.

UDP (*User Datagram Protocol*) est un protocole de connexion discontinue, ce qui signifie qu'il n'y a pas ici création de canal, toutes les données étant transférées au moyen de messages individuels (appelés datagrammes). UDP ne permet pas de connexion sûre et fiable, car les datagrammes peuvent être perdus au cours du transfert.

En général, les différences qui résident entre les modes de livraison de données par les protocoles UDP et TCP sont semblables aux différences entre un appel téléphonique et une carte postale. TCP fonctionne comme un appel téléphonique car il vérifie que la destination est disponible et peut communiquer. UDP fonctionne comme une carte postale ; les messages sont petits et la livraison n'est pas toujours assurée.

UDP est généralement utilisé par des programmes qui transmettent de petites quantités de données en une fois ou présentent des exigences en temps réel. Dans ces cas de figure, les fonctions de faible délai et de multidiffusion du protocole UDP (par exemple, un datagramme, de nombreux destinataires) sont plus adaptées que TCP.

UDP contraste directement avec les services et les fonctionnalités proposés par TCP. Le tableau suivant établit une comparaison des différents modes de gestion des communications TCP/IP suivant que le protocole UDP ou TCP soit utilisé ou non pour le transport des données.

TCP	UDP
Service orienté connexion ; une session est établie entre les hôtes	Service sans connexion ; aucune session n'est établie entre les hôtes
TCP garantit la livraison à l'aide des accusés de réception et la livraison de données en séquence	UDP ne garantit ou n'accuse pas réception des données de livraison ou de séquence
Les programmes qui utilisent TCP sont assurés de la fiabilité du transport des données	Les programmes qui utilisent le protocole UDP doivent fournir une stabilité nécessaire pour le transport des données
TCP est plus lent, présente des exigences de délai plus élevé et ne peut prendre en charge que les communications point à point	UDP est rapide, présente des exigences de faible délai et peut prendre en charge des communications point à point et point à multipoint

1. Identification des connexions

Le TP est réalisé sur des stations Sun disposant de la pile protocolaire du modèle TCP/IP. Solaris offre un certain nombre d'outils, donc un audit réseau **snoop** que nous allons principalement utiliser pour l'analyse du trafic TCP et UDP.

Nous allons mettre en évidence les différentes phases du protocole (connexion, transfert de données, déconnexion) en analysant les TPDUs qui transitent sur le réseau. Pour générer du trafic dans les deux modes, nous pourrons transférer des fichiers, faire une requête DNS, ou même utiliser la commande **ping**. Ces différents types de services sont listés dans le fichier */etc/services*, avec le protocole associé.

Pour analyser et comprendre ce phénomène, nous allons utiliser les applications suivantes :

- *requête ICMP (**ping**)*
- *terminal à distance TELNET*
- *requête DNS*

1.1. Requête ICMP

Le protocole ICMP (*Internet Control Message Protocol*) est un protocole qui permet de gérer les informations relatives aux erreurs des machines connectées. Etant donné le peu de contrôles que le protocole IP réalise, il permet non pas de corriger ces erreurs mais de faire part de ces erreurs aux protocoles des couches voisines. Ainsi, le protocole ICMP est utilisé par tous les routeurs, qui l'utilisent pour signaler une erreur (appelée *Delivery Problem*).

Ce type de requête est utilisé par la commande **ping** ; elle permet notamment de voir si un ordinateur (identifié par son nom d'hôte ou son adresse IP) est connecté au réseau. A l'aide de la commande **snoop –ta sunus5_gi_s05 > ping.txt**, nous pouvons observer ce qu'il se passe sur le réseau lors d'une telle requête au sein d'un fichier texte dont voici un extrait :

```
sunus5-gi-s05 -> sunus5-gi-s08.utbm.fr ICMP Echo request (ID: 2905 Sequence number: 0)
sunus5-gi-s08.utbm.fr -> sunus5-gi-s05 ICMP Echo reply (ID: 2905 Sequence number: 0)
sunus5-gi-s05 -> sunus5-gi-s08.utbm.fr ICMP Echo request (ID: 2906 Sequence number: 0)
sunus5-gi-s08.utbm.fr -> sunus5-gi-s05 ICMP Echo reply (ID: 2906 Sequence number: 0)
sunus5-gi-s05 -> sunus5-gi-s08.utbm.fr ICMP Echo request (ID: 2907 Sequence number: 0)
sunus5-gi-s08.utbm.fr -> sunus5-gi-s05 ICMP Echo reply (ID: 2907 Sequence number: 0)
```

Il s'agit en fait de trois requêtes successives avec leur réponse, toutes réalisées vers la même machine *sunus5_gi_s08* depuis la machine *sunus5_gi_s05*. Le fonctionnement est simple ; chaque ligne indique le sens de communication, ainsi que les machines concernées. Dans le cas présent, nous observons que pour un « aller-retour » (requête + réponse), le numéro ID du datagramme ICMP ne change pas. La machine *sunus5_gi_s05* envoie une requête vers la machine *sunus5_gi_s08* pour connaître son état ; elle est active et connectée, donc elle envoie à son retour une réponse vers *sunus5_gi_s05*.

1.2. Requête DNS

Chaque station possède une adresse IP propre. Cependant, les utilisateurs ne veulent pas travailler avec des adresses numériques du genre *194.153.205.26* mais avec des noms de stations ou des adresses plus explicites (appelées adresses FQDN) du style *http://www.utbm.fr* ou *admin@utbm.fr*. Ainsi, TCP/IP permet d'associer des noms en langage courant aux adresses numériques grâce à un système appelé DNS (*Domain Name Service*). On appelle *résolution de noms de domaines* (ou *résolution d'adresses*) la corrélation entre les adresses IP et le nom de domaine associé.

Dans le cas présent, les requêtes DNS s'effectuent constamment sur la station concernée, il est donc aisé de les identifier. Prenons par exemple les échanges suivants :

```
sunus5-gi-s05 -> pc-cri-b08.utbm.fr DNS C com-res-08.utbm.fr. Internet Addr ?
pc-cri-b08.utbm.fr -> sunus5-gi-s05 DNS R com-res-08.utbm.fr. Internet Addr 10.0.0.3
sunus5-gi-s05 -> pc-cri-b08.utbm.fr DNS C pc-cri-b08.utbm.fr. Internet Addr ?
pc-cri-b08.utbm.fr -> sunus5-gi-s05 DNS R pc-cri-b08.utbm.fr. Internet Addr 10.0.0.25
```

Le mécanisme est analogue à celui de la requête ICMP :

- *le client* sunus5_gi_s05 *demande au serveur* pc_cri_b08.utbm.fr *l'adresse IP de la machine* com_res_08.utbm.fr
- *le serveur* pc_cri_b08.utbm.fr *donne l'adresse IP de la machine concernée :* 10.0.0.3, *soit une adresse IP locale de classe A*

On apprend aussi que l'adresse IP du serveur *pc_cri_b08.utbm.fr* est *10.0.0.25*. A noter que la commande **ping** permet d'effectuer cette opération (DNS > IP) mais pas l'inverse.

1.3. Terminal à distance Telnet

Le protocole Telnet est un protocole standard d'Internet permettant l'interfaçage de terminaux et d'applications à travers Internet. Ce protocole fournit les règles de base pour permettre de relier un client (système composé d'un affichage et d'un clavier) à un interpréteur de commande (côté serveur). Telnet s'appuie sur une connexion TCP pour envoyer des données au format ASCII codées sur 8 bits entre lesquelles s'intercalent des séquences de contrôle Telnet. Il fournit ainsi un système orienté communication, bi-directionnel (half-duplex), codé sur 8 bits facile à mettre en oeuvre.

Contrairement aux protocoles précédents, la mise en place d'une communication est un peu plus complexe. Le principe est le suivant ; la machine voulant se connecter sur une autre machine envoie une requête de connexion :

```
sunus5-gi-s05 -> 10.90.0.229  TELNET C port=32814
10.90.0.229 -> sunus5-gi-s05 TELNET R port=32814 Last login: Wed Oct 13 15:12:00
sunus5-gi-s05 -> 10.90.0.229  TELNET C port=32814
```

Sur chaque ligne nous pouvons voir les stations concernées, mais aussi le port source utilisé pour la connexion (*32814*), sachant que le port destination utilisé est *23* par convention.

La station *sunus5_gi_s05* envoie une demande de connexion à la machine *sunus5_gi_s09* d'adresse *10.90.0.229*. Une fois la connexion établie, *sunus5_gi_s09* informe *sunus5_gi_s05* de la date de la dernière connexion (*Last login*).

Pour un transfert de données, le fonctionnement est analogue :

```
sunus5-gi-s05 -> 10.90.0.229  TELNET C port=32814  d
10.90.0.229 -> sunus5-gi-s05 TELNET R port=32814 d
10.90.0.229 -> sunus5-gi-s05 TELNET R port=32814 /home/invite\r\n$
sunus5-gi-s05 -> 10.90.0.229  TELNET C port=32814
```

A chaque donnée transmise, *sunus5_gi_s09* envoie une confirmation (*acknowledgement*) ; la réponse (*/home/invite\r\n$*) est envoyée à *sunus5_gi_s05*.

Remarque

La commande **netstat** permet d'afficher les statistiques de protocole et les connexions réseau TCP/IP en cours. Elle peut donc présenter une alternative ou un complément utile à **snoop** pour le travail effectué précédemment. Néanmoins, certaines options sont nécessaires pour améliorer la « lisibilité » des résultats.

2. Transmission – Retransmission des données

2.1. Transmission réelle de données

Pour mettre en évidence les phénomènes demandés, nous avons utilisé un transfert de fichier par le protocole FTP (File Transfert Protocol). Il définit la façon selon laquelle des données doivent être transférées sur un réseau TCP/IP. Il a pour objectif de :

- *permettre un partage de fichiers entre machine distante*
- *permettre une indépendance aux systèmes de fichiers des machines clientes et serveur*
- *permettre de transférer des données de manière efficace*

Nous avons tout d'abord réalisé 2 transmissions réelles en utilisant les commandes FTP entre deux postes de la salle de TP, à savoir les postes *sunus5-gi-s05* et *sunus5-gi-s09*.

Le principe consiste à envoyer la même quantité de données de deux manières différentes, une fois en 10 petits fichiers et la deuxième fois en un seul fichier de la taille des 10 petits fichiers cumulés.

Pour cela nous avons donc créé un fichier de 10 Mo de texte (une simple suite de « 0 ») via l'éditeur **vi** de la manière suivante :

```
# !/bin/sh
I=0
Data = 0
While [$i < $1]
Do
File = $file$data
I = 'expr $i + 1'
Done
echo $ file
```

- *pour le rendre exécutable :* ***chmode + x***
- *pour créer un fichier de 1 ko :* ***make-file.sh 1024***
- *pour doubler le fichier successivement jusqu'à obtenir la taille désirée :*
  cat file1.data file1.data > file2.data

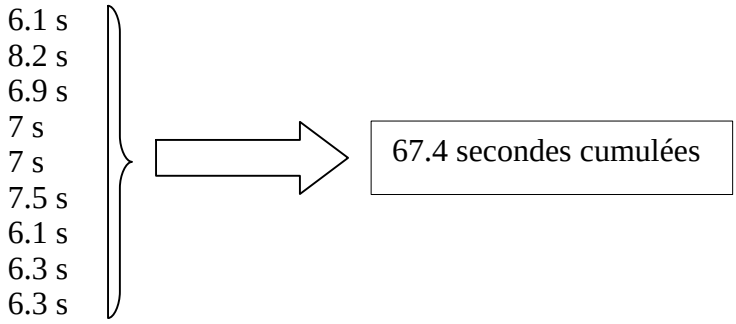
Nous avions ainsi un fichier de 10 Mo et un autre de 100 Mo rempli de la même chaîne de caractères. Il était essentiel par la suite d'afficher l'heure dans l'audit de réseau **snoop** afin que la saisie du mot de passe lors du login sur la machine de destination ne fausse pas les résultats :

snoop –t a | grep sunus5-gi-S09

Nous pouvions ensuite procéder à notre batterie de tests en envoyant nos fichiers dans un répertoire *data* de la machine de destination :

rcp file10M.data « sunus5-gi-S05 : ~/data/file$i.data »

Ainsi nous avons pu observer les temps de transferts suivants pour le premier fichier de 10 Mo :



Pour le fichier de 100 Mo nous avons réalisé deux transferts successifs. Ils ont tous les deux donnés le résultat suivant : 44 secondes pour le transfert complet soit près de 40 % de temps en moins par rapport à la première méthode.

Ces résultats ne sont évidemment pas surprenants. La raison d'une telle différence s'explique par les entêtes et les champs de contrôle nécessaires au transfert de chaque paquet, qui sont évidemment 10 fois plus nombreux dans la première méthode car indépendante de la taille du fichier à transmettre dans notre cas (10 Mo ou 100 Mo).

Cependant on peut constater que la différence de temps est quand même considérable pour un rapport de 1 à 10. Elle le serait d'autant plus pour un rapport plus grand, c'est donc un paramètre à ne pas négliger.

2.2. Transmission avec incident

Enumérer les différents TPDUs d'un protocole ne suffit pas pour comprendre totalement son fonctionnement. Certains mécanismes sont observables grâce à la réalisation d'autres mesures. Pour TCP, on peut facilement observer deux types de comportements : le renvoi de paquets après occurrence d'une panne ou la corruption des données (données erronées), et la négociation de la MSS (*Maximum Segment Size*).

Pour provoquer cette situation de panne, il suffit de débrancher le câble réseau d'une des deux machines pendant quelques secondes pendant le transfert d'un fichier de taille suffisante pour l'exploitation des résultats.

A la fin du transfert il nous suffisait donc de lire le rapport de transfert pour chaque paquet. Celui qui n'était pas passé du premier coup et qui avait été retransmis sans cesse pendant un certain temps nous indiquait donc le moment où nous avions débranché le câble réseau.

Pour chaque paquet transmis la machine réceptrice devait renvoyer des accusés de réception visible par le numéro *acknowledgement*. Le sens de la transmission pouvait être

déterminé en regardant les adresses d'émission des cartes réseaux (*source address* et *destination address*).

Nous avons comparé les temps de transmission d'un fichier de taille 1 Mo dans deux cas :

- *transmission sans panne*
- *transmission avec panne*

Puis nous avons obtenu les résultats suivants :

```
200 PORT command successful.
150 ASCII data connection for core (10.90.0.229,32881) (1040992 bytes).
226 ASCII Transfer complete.
local: testftp remote: core
1044037 bytes received in 20.1 seconds (140.97Kbytes/s)

200 PORT command successful.
150 ASCII data connection for core (10.90.0.229,32881) (1040992 bytes).
226 ASCII Transfer complete.
local: testftp remote: core
1044037 bytes received in 511.8 seconds (52.04Kbytes/s)
```

Le résultat est logique ; le temps de transfert augmente largement pendant que le débit chute d'autant. Ceci est dû certes au temps de panne pendant lequel aucune donnée ne transite, mais aussi à la retransmission des données envoyées pendant la panne, ce qui constitue un délai supplémentaire. Lors de la coupure il a donc fallu plusieurs dizaines de secondes avant la récupération de transfert. De même on a pu observer une baisse du nombre de trames reçues avant un acquittement, en pratique ce nombre est divisé par 2. Nous pouvons par conséquent supposer une certaine adaptation des transmissions à la qualité du réseau, ce qui explique l'importante dégradation du débit.

En effet, lorsque que la communication est interrompue de manière physique, la machine source continue d'envoyer des requêtes à la machine destination sans recevoir les paquets *ACK*. A ce moment-là, le *timeout* (temps maximal d'attente d'une trame de confirmation) augmente progressivement jusqu'à ce que la machine recommence à recevoir les trames *ACK*. Ainsi, les trames transmises mais non confirmées sont réexpédiées, et le *timeout* diminue progressivement jusqu'à sa valeur initiale.

3. Différence de comportement des réseaux

Le MSS (*Maximum Segment Size*) permet à une machine source de spécifier une taille maximale de segment analogue au MTU (*Maximum Transmission Unit*), taille maximale d'une trame physique sur un réseau.

Ainsi, à l'ouverture d'une session TCP, chaque machine peut annoncer son MSS à l'autre, sachant que celui-ci vaut 536 octets par défaut et qu'il se calcule en fonction du MTU :

MSS = MTU – 40 octets

Ceci permet par exemple de « fiabiliser » la transmission de paquets sur un réseau en diminuant le MTU/MSS ; dans ce cas plus de paquets sont transmis et le rendement (débit utile / débit réel) est moindre car le débit réel est augmenté. Dans le cas contraire, moins de paquets sont envoyés, le rendement augmente, mais la fiabilité du support réseau doit être meilleure pour éviter de réexpédier des paquets qui deviennent de fait plus volumineux.

Cela se vérifie en pratique.

Lorsque nous procédons au transfert d'un fichier d'une taille quelconque par le réseau local (comme vu précédemment), nous observons, grâce à la commande **snoop –v**, un MSS de valeur 1460 octets. Ce résultat est tout à fait cohérent avec la valeur nominale du MTU en Ethernet, à savoir 1500 octets (1500 – 40 = 1460 octets).

Ensuite, lors d'un transfert depuis un serveur FTP accessible via Internet, la valeur constatée du MSS tombe à 536 octets (soit un MTU de 576 octets), soit une taille de segment bien inférieure à celle constatée dans un transfert sur un réseau local.

Ces deux transferts nous ont permis de bien mettre en évidence d'une part problème de négociation du MSS, et d'autre part sa variation en fonction du type de réseau utilisé, qu'il soit global (Internet) ou local (Ethernet).