Antoine Foucault
Réseau
26/11/2013

Rapport du TP n°2

Sommaire

1 Introduction

L'objectif de ce TP sera de comprendre de façon rudimentaire comment un système d'exploitation Linux gère les protocoles réseaux et sa configuration réseau.

La deuxième partie consistera quand à elle à prendre en main les outils de programmation du langage C afin de faire communiquer différents programmes via l'utilisation de sockets TCP et UDP.

2 Recherche d'informations

Lors du début de ce TP nous avons examiné le contenu de certains fichiers et dossiers contenus dans le dossier “/etc” d'un système Linux dont la description se situe dans les chapitres ci-dessous.

2.1 Fichier “protocols”

Le fichier “/etc/protocols” definit l'ensemble des sous-protocoles de TCP/IP disponibles sur le système.

Le fichier se présente sous cette forme : **[nom de protocole] [numéro] [alias].**

Le numéro de protocole présent est définit par le DDN Network Information Center qui est une autorité entre autre chargée de définir un numéro unique à chaque protocole qui le nécessite.

C'est ce numéro qu'on retrouve dans le champs protocole de l'entête IP lors des communications réseau.

Une description de ce fichier est disponible dans l'illustration n°1.

```
# /etc/protocols:
# $Id: protocols,v 1.11 2011/05/03 14:45:40 ovasik Exp $
#
# Internet (IP) protocols
#
#   from: @(#)protocols 5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
# Last IANA update included dated 2011-05-03
#
# See also http://www.iana.org/assignments/protocol-numbers

ip  0   IP      # internet protocol, pseudo protocol number
hopopt  0   HOPOPT      # hop-by-hop options for ipv6
icmp    1   ICMP       # internet control message protocol
igmp    2   IGMP       # internet group management protocol
ggp 3   GGP     # gateway-gateway protocol
ipv4    4   IPv4        # IPv4 encapsulation
st  5   ST      # ST datagram mode
tcp 6   TCP     # transmission control protocol
cbt 7   CBT     # CBT, Tony Ballardie <A.Ballardie@cs.ucl.ac.uk>
egp 8   EGP     # exterior gateway protocol
igp 9   IGP     # any private interior gateway (Cisco: for IGRP)
bbn-rcc 10  BBN-RCC-MON     # BBN RCC Monitoring
nvp 11  NVP-II      # Network Voice Protocol
pup 12  PUP     # PARC universal packet protocol
argus   13  ARGUS       # ARGUS
emcon   14  EMCON       # EMCON
xnet    15  XNET       # Cross Net Debugger
chaos   16  CHAOS       # Chaos
udp 17  UDP     # user datagram protocol
mux 18  MUX     # Multiplexing protocol
```

Illustration 1 : Le fichier "protocols".

Les noms de protocoles en rouge, les numéros de protocole en vert et les alias en bleu

2.2 Fichier “hosts”

Le fichier “/etc/hosts” définit l'association de certains noms de domaine à certaines adresses IP.

Afin de résoudre une adresse/un nom de domaine, le système commence par consulter le fichier “hosts” avant de consulter une base de donnée et finalement envoyer une requête DNS à un/plusieurs serveurs.

Le fichier “hosts” est organisé selon le schéma suivant : **[Adresse IP] [Nom de domaine] [Alias1]...**

L'illustration n°2 décrit le fichier.

```
127.0.0.1       localhost.localdomain localhost
::1     localhost6.localdomain6 localhost6
```

Illustration 2 : Le fichier "hosts".

En rouge les adresse (ici en IPv4 et IPv6) et en vert les noms de domaine associés

2.3 Fichier “services”

Le fichier “/etc/services” fournit une correspondance entre un service internet et son couple port/protocole d'utilisation.

Le fichier est organisé comme suit : **[nom du service] [port/protocole] [alias1] [aliasN]**

Les numéros de ports sont affectés par l'IANA qui est une autorité chargée d'assigner un port à un service. Sa politique actuelle est d'affecter les protocoles TCP et UDP à chaque service, c'est pourquoi on retrouve beaucoup de pseudo-doublons.

L'illustration n°3 décrit ce fichier.

Illustration 3 : Le fichier "services".

En rouge le service, en bleu le numéro de port associé, en vert le protocole et en jaune les alias.

2.4 Dossier “init.d”

Le dossier init.d contient les scripts de gestion des différents services du système, cela va du service de l'interface graphique (par exemple X11) aux services réseaux comme le HTTP.

Par exemple, lors de l'utilisation de la commande “*services httpd start*” le serveur Apache est démarré via un script dans ce dossier.

2.5 Fichier “nsswitch.conf”

Le fichier “nsswitch.conf” (Name Service Switch) définit l'ordre de consultation des différentes bases de données distantes/du système lors d'une recherche de ces informations.

Le fichier “nsswitch.conf” est organisé comme suit :

[Ressource] [Moyen d'accès prioritaire] [Moyen d'accès un peu moins prioritaire]...

L'illustation n°4 décrit ce fichier.

```
passwd:         files
shadow:         files
group:          files
#initgroups: files

#hosts:     db files nisplus nis dns
hosts:      files mdns4_minimal [NOTFOUND=return] dns myhostname
```

Illustration 4 : Contenu du fichier "nsswitch.conf".

En rouge la ressource à laquelle accéder, en bleu l'ordre des bases de données à consulter.

3 Programmation sockets UDP

La programmation via les sockets UDP consiste à envoyer des datagrammes sans savoir si le récepteur a bien reçu celui-ci ou non. Cela implique donc un “risque” plus important quand à la fiabilité du transfert des données (serveur ou client déconnecté/inexistant) mais permet également des communications moins lourdes et donc plus rapides.

3.1 Reprise du TD sur les sockets UDP

Le TD sur les sockets UDP consistait en deux applications, un client et un serveur chargées de se communiquer entre elles un fichier contenant du texte. Le serveur devait également afficher le paquet reçu et le renvoyer afin que le client vérifie qu'il était identique à celui émit.

La figure 5 illustre le fonctionnement des deux applications.

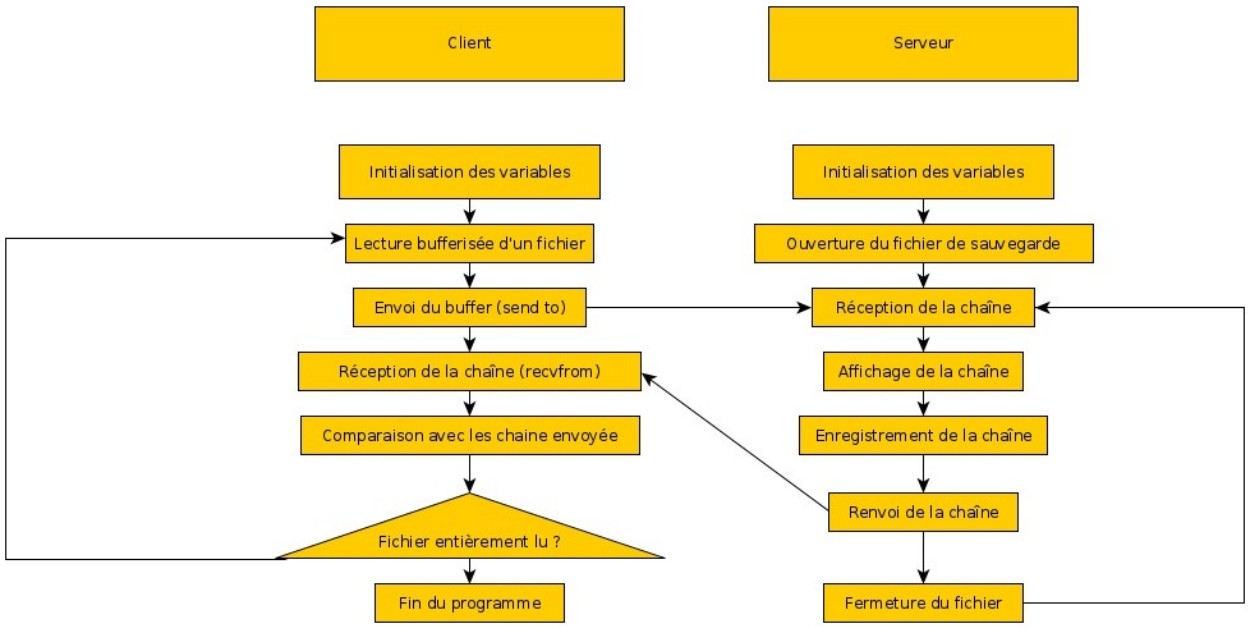


Illustration 5 : Déroulement de l'intéraction entre client et serveur.

3.2 Test de l'application

Nous avons testé l'application en utilisant les applications client/serveur en local et à l'aide de deux pc.

Pour chaque utilisation les tests suivants ont été réalisés :

- Capture de l'échange entre les applications à l'aide du logiciel “Wireshark”
- Comparaison des deux fichiers à l'aide du logiciel “diff”

Les captures sont disponibles dans le dossier “Annexes/A2/Captures” sous le nom “A2(local).pcapng” et “A2(distant).pcapng”.

Le programme est disponible dans le dossier “Annexes/A2/Programme”

a) Capture réseau locale

44 Source port: 43751	Destination port: 24500
44 Source port: 24500	Destination port: 43751
44 Source port: 43751	Destination port: 24500
44 Source port: 24500	Destination port: 43751
44 Source port: 43751	Destination port: 24500
44 Source port: 24500	Destination port: 43751
44 Source port: 43751	Destination port: 24500
44 Source port: 24500	Destination port: 43751
44 Source port: 43751	Destination port: 24500
44 Source port: 24500	Destination port: 43751

b) Capture réseau distante

Pour la capture distante on a utilisé le filtre de capture “udp port 24500”.

192.168.61.152	192.168.61.12	UDP	44 Source port: 55554	Destination port: 24500
192.168.61.12	192.168.61.152	UDP	60 Source port: 24500	Destination port: 55554
192.168.61.152	192.168.61.12	UDP	44 Source port: 55554	Destination port: 24500
192.168.61.12	192.168.61.152	UDP	60 Source port: 24500	Destination port: 55554
192.168.61.152	192.168.61.12	UDP	44 Source port: 55554	Destination port: 24500
192.168.61.12	192.168.61.152	UDP	60 Source port: 24500	Destination port: 55554
192.168.61.152	192.168.61.12	UDP	44 Source port: 55554	Destination port: 24500
192.168.61.12	192.168.61.152	UDP	60 Source port: 24500	Destination port: 55554

On observe une alternance entre les envoi et les retours des paquets.

c) Comparaison des fichiers

Les tests de comparaison des fichiers effectués avec le logiciel “diff” n'ont montré aucune différence entre les fichiers téléchargés/téléversés (uploadés) et les fichiers originaux.

3.3 Transfert de taille importante

Lors d'un transfert de taille importante on constate que le programme effectue correctement son transfert.

4 Programmation sockets TCP

La programmation via les sockets TCP requiert d'abord une connexion du client au serveur avant l'échange de données (le three-way handshake). Le protocole TCP essaie d'assurer l'arrivée d'un paquet à sa destination par le renvoi des données par l'émetteur en cas d'erreur dans l'intégrité du paquet.

De ce fait le protocole TCP est considéré comme plus fiable que le protocole UDP. Cependant il est également plus lourd à cause de l'ajout d'informations permettant de vérifier l'intégrité d'un paquet ainsi que le système de vérification de l'arrivée des paquets.

4.1 Reprise du TD sur les sockets TCP

Le premier exercice a consisté à programmer un client et un serveur chargés :

- Pour le client d'envoyer un fichier au serveur via un socket TCP.
- Pour le serveur de sauvegarder le fichier reçu via le réseau dans le dossier "/tmp" du système.

4.1.1 Organisation de l'application

L'illustration 6 résume le fonctionnement de l'application.

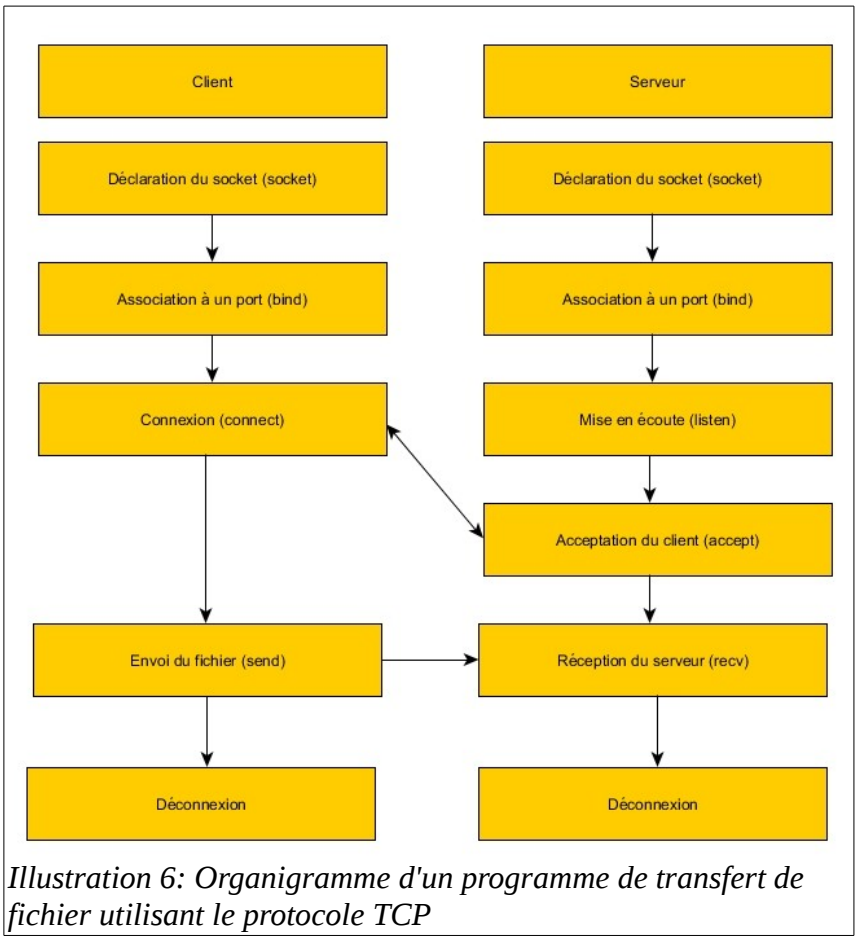


Illustration 6: Organigramme d'un programme de transfert de fichier utilisant le protocole TCP

4.1.2 Analyse

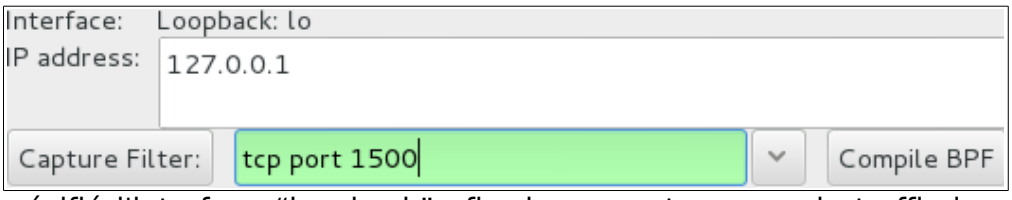
Afin de vérifier que l'application fonctionne bien on a comparé les fichiers envoyés et reçus à l'aide du logiciel “diff” (chargé de détecter les différences entre deux fichiers) qui n'a détecté aucune différence entre le fichier envoyé et reçu de quelque taille qui soit.

On a ensuite observé les échanges réseaux des deux applications à l'aide de l'analyseur de trames réseau Wireshark. Les fichiers de capture des échanges entre les deux application sont disponible dans le dossier “Annexes/A3a8/Captures” sous le nom “A3a8(local).pcapng” et “A3a8(distant).pcapng”.

L'application est disponible dans le dossier “Annexes/A3a8/Programme”

Voici la **configuration de Wireshark** pour les captures :

- Filtre pour la capture locale



On a spécifié l'interface “loopback” afin de ne capturer que le traffic local sur le port 1500

- Filtre pour la capture distante

On filtre le port 53 pour le traffic DNS et on filtre le traffic sur le port TCP 1500 avec l'hôte 192.168.61.12.

a) Capture réseau locale

L'illustration 7 représente un extrait de la capture réseau effectuée lors du test de nos programmes en local.

```
50329 > vlsi-lm [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK
vlsi-lm > 50329 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS
50329 > vlsi-lm [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=77
50329 > vlsi-lm [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=100 T
vlsi-lm > 50329 [ACK] Seq=1 Ack=101 Win=43776 Len=0 TSval=
50329 > vlsi-lm [PSH, ACK] Seq=101 Ack=1 Win=43776 Len=18
vlsi-lm > 50329 [ACK] Seq=1 Ack=119 Win=43776 Len=0 TSval=
50329 > vlsi-lm [FIN, ACK] Seq=119 Ack=1 Win=43776 Len=0 T
vlsi-lm > 50329 [FIN, ACK] Seq=1 Ack=120 Win=43776 Len=0 T
50329 > vlsi-lm [ACK] Seq=120 Ack=2 Win=43776 Len=0 TSval=
```

Illustration 7: Extrait de la capture réseau locale.

On remarque encouleur rouge le 3-way handshake de connexion composé :

- D'un segment SYN envoyé par le client au serveur
- D'un segment SYN-ACK renvoyé par le serveur au client
- De la confirmation du client par un ACK au serveur

En bleu on peut voir la déconnexion en 3 temps composé :

- D'un segment FIN_ACK envoyé par le client
- D'une réponse FIN_ACK du serveur au client
- D'une confirmation par un ACK du client au serveur

En jaune on peut observer la taille de nos données encapsulées par le paquet TCP, le premier paquet envoyé à une taille de 100 octets, correspondant à la taille max. du buffer de nos programme, puis le second paquet contient 18 octets correspondant à la fin des données à transmettre.

On pourra également noter l'évolution des ACK (acquittements) et SEQ (sequencements) des deux applications au cours du transfert ainsi que l'utilisation des port 50329 pour le client et 1500 pour le serveur (vlsi-lm).

Le ACK d'une machine est incrémenté par la taille des données encapsulées envoyées par l'autre application. Une fois les données reçues le SEQ de l'application emettrice se positionne à cette valeur et ainsi de suite.

Ce système de TCP permet à une application de remettre les paquets dans l'ordre et également d'assurer à une application que les données qu'elles a reçu précédemment sont bien traités par l'application réceptrice.

b) Capture réseau distante

L'illustration 8 représente un extrait de la capture réseau effectuée lors de l'utilisation de nos deux programmes avec 2 machines distinctes connectées en réseau.

192.168.61.152	192.168.61.250	DNS	67 Standard query 0xdf8e A bt.home
192.168.61.250	192.168.61.152	DNS	83 Standard query response 0xdf8e A 192.168.61.12
192.168.61.152	192.168.61.12	TCP	74 46553 > vlsi-lm [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=
192.168.61.12	192.168.61.152	TCP	74 vlsi-lm > 46553 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460
192.168.61.152	192.168.61.12	TCP	66 46553 > vlsi-lm [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=9739663
192.168.61.152	192.168.61.12	TCP	166 46553 > vlsi-lm [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=100 TSval=
192.168.61.152	192.168.61.12	TCP	84 46553 > vlsi-lm [FIN, PSH, ACK] Seq=101 Ack=1 Win=29312 Len=18
192.168.61.12	192.168.61.152	TCP	66 vlsi-lm > 46553 [ACK] Seq=1 Ack=101 Win=14480 Len=0 TSval=62884
192.168.61.12	192.168.61.152	TCP	66 vlsi-lm > 46553 [FIN, ACK] Seq=1 Ack=120 Win=14480 Len=0 TSval=
192.168.61.152	192.168.61.12	TCP	66 46553 > vlsi-lm [ACK] Seq=120 Ack=2 Win=29312 Len=0 TSval=97396

Illustration 8: Extrait de la capture réseau distante

On remarque en rouge l'envoyeur du paquet, en bleu on peut voir le destinataire du paquet. En vert on a le protocole filtré.

On remarque d'abord un jeu de requêtes DNS, en effet le client émet une requête DNS (en demandant bt.home) au serveur DNS (ici la passerelle) qui lui répond avec l'adresse 192.168.61.12. Les échanges qui suivent concernent le client et le serveur. On remarque

d'ailleurs en jaune la taille de notre fichier qui est divisé en deux paquets.

4.2 Application “transfert de nombres aléatoires”

Le second exercice a consisté à programmer un client et un serveur chargés :

- Pour le client de se connecter à un serveur et de récupérer 10 nombres
- Pour le serveur d'envoyer à la connexion d'un client 10 nombres aléatoires

4.2.1 Organisation de l'application

L'illustration 9 resume le fonctionnement de l'application.

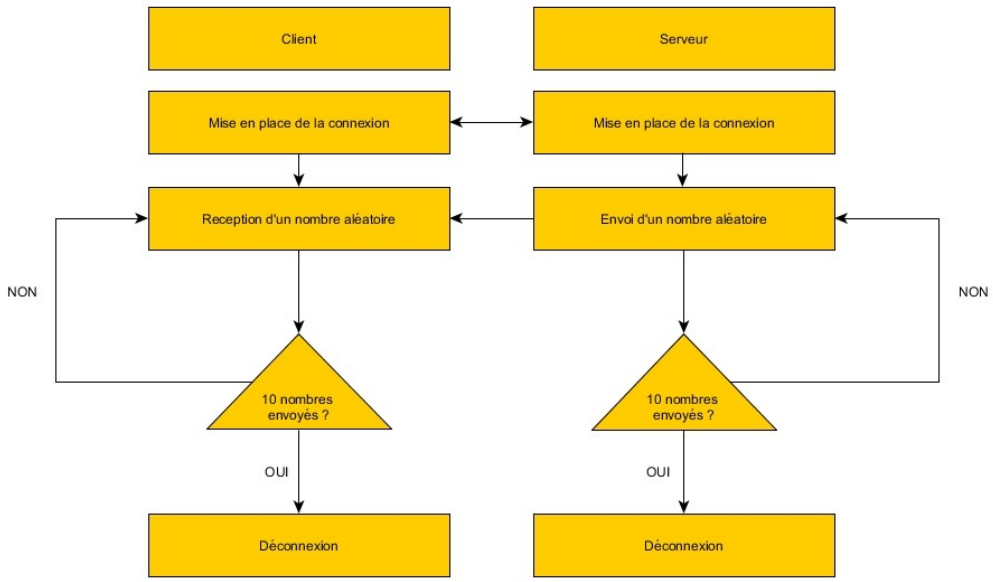


Illustration 9: Organigramme de deux applications chargées de se transférer des nombres aléatoires.

Les étapes de connexion étant les mêmes que celles décrites dans l'organigramme précedent, elles sont résumées par la partie “Mise en place de la connexion”

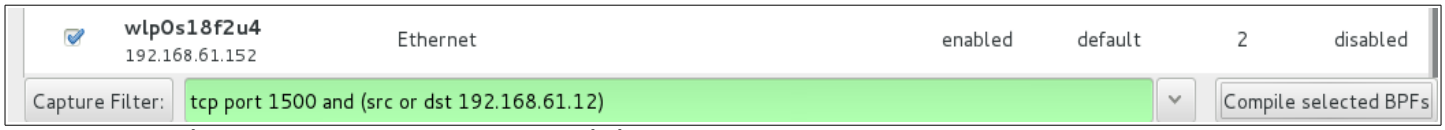
4.2.2 Analyse

L'analyse du fonctionnement de nos applications n'a cette fois pas nécessité de comparer deux fichiers. Nous nous sommes donc seulement contenté d'analyser le traffic réseau entre nos deux applications à l'aide du logiciel Wireshark.

Le fichier de capture est disponible dans le dossier “Annexes/A3a10/Capture” sous le nom “A3a10.pcapng”.

Les sources du programme sont disponibles dans le dossier “Annexes/A3a10/Programme”.

Voici la **configuration de Wireshark** pour la capture :



Le système distant est donc situé à l'adresse *192.168.61.12 et on utilise l'interface "wlp0s18f2u4" dont l'adresse IP est 192.168.61.152. On a donc les deux machines situées sur le même réseau.*

a) Capture réseau distante

L'illustration 10 présente un extrait de la capture réseau distante.

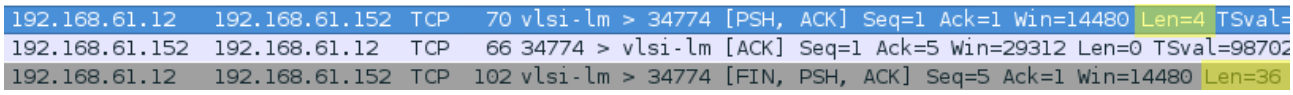


Illustration 10: Extrait de la capture distante

On remarque les même champs que dans les captures précédentes. On note également que l'addition de longueur des différents paquets vaut 40 octets ce qui correspont à 4*10 int. Les deux applications échangent donc bien 10 nombres aléatoires.

4.3 Application "ServeurEcho"

Le troisième exercice a consisté à programmer un client et un serveur chargés :

- Pour le client de se connecter à un serveur d'envoyer une réquête et de calculer le temps de tranfert de cette requête-réponse.
- Pour le serveur de répondre à la requête d'un client par cette même requête.

4.3.1 Organisation de l'application

L'illustration 11 resume le fonctionnement de l'application.

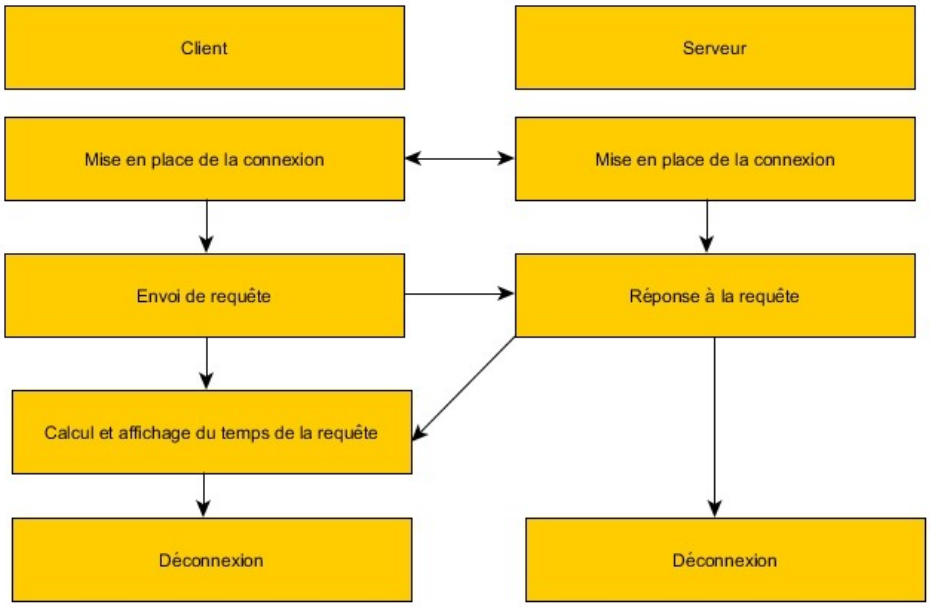


Illustration 11: Organigramme de deux applications client-serveur.

Le client est chargé de déterminer le temps de connexion avec l'autre machine

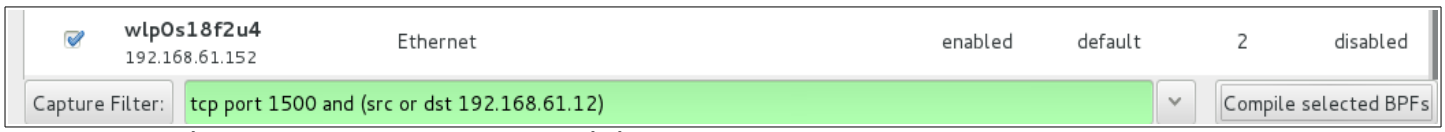
4.3.2 Analyse

L'analyse du fonctionnement de nos applications n'a encore cette fois pas nécessité de comparer deux fichiers. Nous nous sommes donc seulement contenté d'analyser le traffic réseau entre nos deux applications à l'aide du logiciel Wireshark.

Le fichier de capture est disponible dans le dossier “Annexes/A3a11/Capture” sous le nom “A3a11.pcapng”.

Les sources du programme sont disponibles dans le dossier “Annexes/A3a11/Programme”.

Voici la **configuration de Wireshark** pour la capture :



Le système distant est donc situé à l'adresse *192.168.61.12 et on utilise l'interface “wlp0s18f2u4” dont l'adresse IP est 192.168.61.152. On a donc les deux machines situées sur le même réseau.*

a) Capture réseau distante

L'illustration 12 illustre l'échange de paquet entre les deux applications afin de déterminer le temps de connection (échange d'un int).

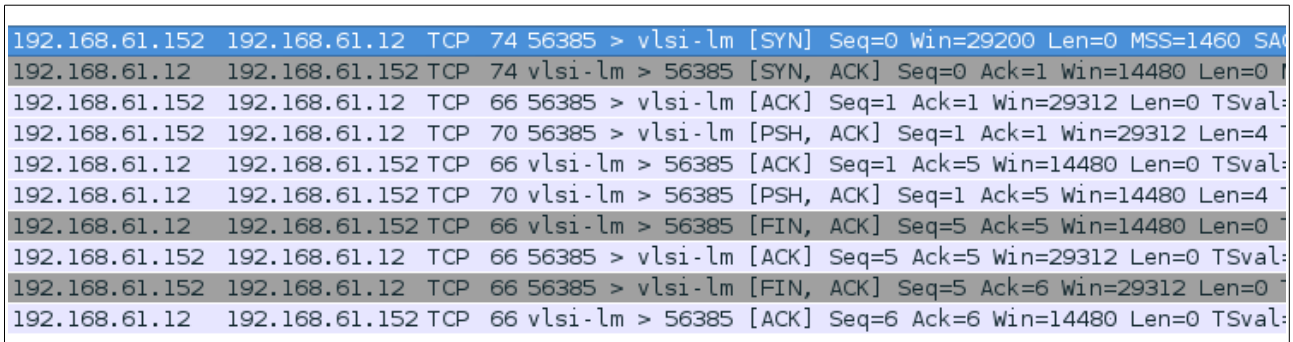


Illustration 12: Echange d'un int entre les deux applications.

4.4 Programmation d'un mini-serveur FTP

Le dernier exercice de ce TP consistait à programmer un mini serveur-FTP et son client capables d'effectuer les opérations suivantes :

- Serveur
  - Lister le contenu du répertoir de travail du serveur (commande “ls”)
  - Se déplacer dans l'arborescence du serveur (commande “cd”)
  - Tranférer un fichier du serveur vers le client (commande “get”)
  - Transférer un fichier du client vers le serveur (commande “put”)
  - De terminer la connexion du client et du serveur (commande “exit”)
- Client
  - Lister le contenu du répertoire local du client (commande “lls”)
  - Se déplacer dans l'arborescence du client (commande “lcd”)

Il a été choisi d'utiliser le port 21 pour les connexions sur le serveur, ce port étant connu pour gérer le traffic FTP.

4.4.1 Identification des structures de données

On a choisi de créer 2 structures de données pour réaliser les échanges entre le client et le serveur.

Le contenu de ces structures est dans le tableau ci-dessous.

Requête	Réponse
Commande Taille de la transmission Nom de fichier	Code de retour Taille de la transmission

Le champs “commande” de la structure requête contiendra la commande que le client souhaite effectuer sur le serveur.

Les champs “Nom de fichier” et “Taille de la transmission” donnent toutes le informations nécessaires au serveur lorsque le client souhaite téléverser (uploader) un fichier dessus.

Le champs “Code retour” de la structure de réponse sert à envoyer un statut du traitement de la commande par le serveur au client définit comme tel :

- Code n°**500** : Erreur dans le traitement de la commande, message d'erreur de la taille “Taille de la transmission” transmit dans le prochain envoi du serveur.
- Code n°**200** : Traitement de la commande correct, poursuite du traitement en fonction de la commande.

4.4.2 Organigramme

L'illustration 13 décrit le fonctionnement du client et du serveur.

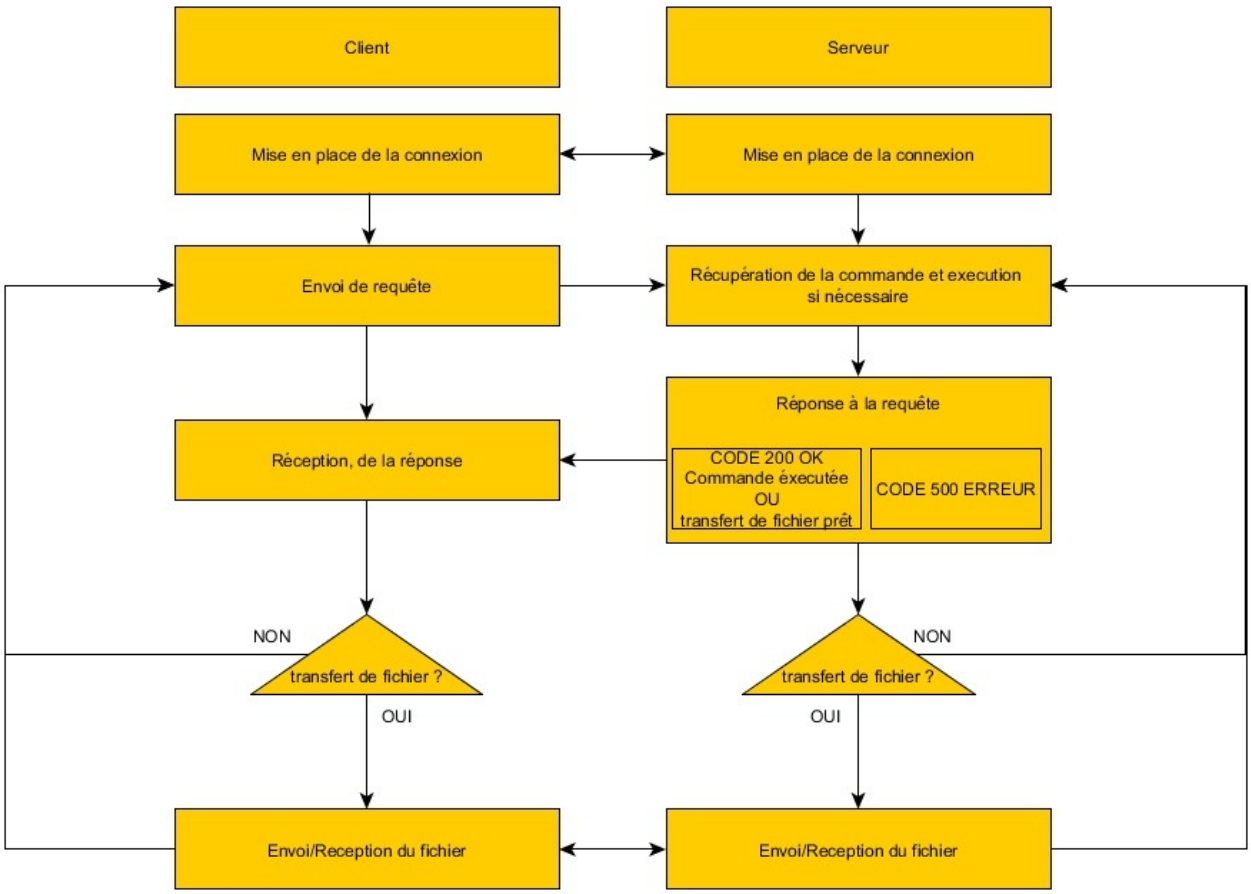


Illustration 13: Organigramme de l'intéraction entre le client et le serveur FTP

4.4.3 Tests et analyse

Les tests on consisté en :

- La capture de l'échange entre les applications à l'aide du logiciel “WireShark”
- La comparaison des deux fichiers originaux à ceux téléchargés ou téléversés (uploadés) à l'aide du logiciel “diff”

Le fichier de capture est disponible dans le dossier “Annexes/A3b/Capture” sous le nom “A3b.pcapng”.
Les sources du programme sont disponibles dans le dossier “Annexes/A3b/Programme”.

Voici la **configuration de Wireshark** pour la capture :

Le système distant est donc situé à l'adresse *192.168.61.12 et on utilise l'interface “wlp0s18f2u4” dont l'adresse IP est 192.168.61.152. On a donc les deux machines situées sur le même réseau.*

a) Capture réseau distante

L'illustration 14 représente un extrait de la capture réseau des échanges entre le client et le serveur.

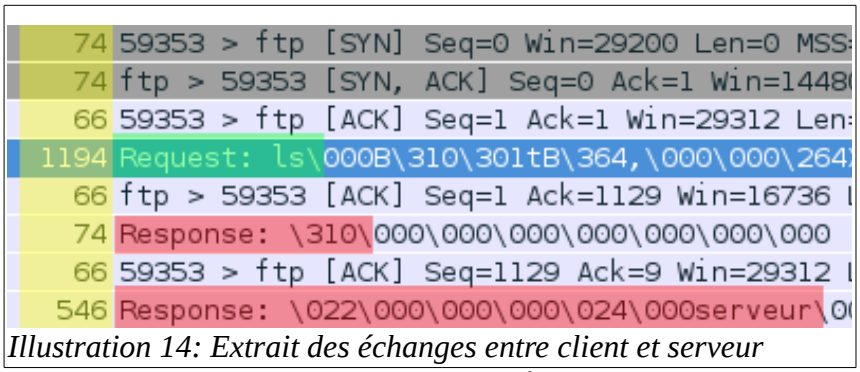


Illustration 14: Extrait des échanges entre client et serveur

On peut voir en jaune la taille des paquets envoyés, en vert le “contenu” des requêtes et en rouge le “contenu” des réponses.

On remarque ici que le client effectue une requête “ls”.

Le serveur lui répond avec la taille des données à suivre puis envoie le premier fichier nommé “serveur” et ainsi de suite.

b) Comparaison des fichiers

Les tests de comparaison des fichiers effectués avec le logiciel “diff” n'ont montré aucune différence entre les fichiers téléchargés/téléversés (uploadés) et les fichiers originaux.

5 Conclusion

Ce TP nous as permis de comprendre une partie des mécanismes mis en place par les systèmes Linux afin de paramétrer un ordinateur pour fonctionner avec les composants et l'organisation unique de chaque réseau.

La deuxième partie a été consacrée à la programmation réseau en C ainsi que l'analyse des trames de nos programmes afin de faire le lien entre nos données envoyées par les programmes et leurs représentation “brute” au niveau du réseau.