Episode 5 - Applications réseaux simples

Programmation système unix

BOOTSTRAP

Applications réseaux simples

Il est temps d'utiliser la puissance du réseau.
A partir de maintenant, nous allons aborder la création d'applications qui communiquent au travers du réseau TCP/IP.

Ce que nous allons voir est à la base de 99% des logiciels et applications que nous utilisons au quotidien : une architecture client / serveur en TCP.

Faites bon usage de cette partie.

Même si vous êtes un full system & network guy, tous les outils et logiciels que vous utilisez implémentent les concepts que nous allons développer ici. Cela peut vous être grandement utile pour comprendre comment ça fonctionne "sous le capot".

PREAMBULE

Pour bien commencer votre journée, voici quelques questions très simples :

- Que se passerait-il si je laissais allumé un sèche-cheveux alimenté en continu dans un cube étanche d'un mètre de côté ?

- Déverser de l'antimatière dans le réacteur de Tchernobyl quand il était en train de fondre aurait-il empêché sa fusion ?

- C'est possible de pleurer au point de se déshydrater ?

- Si tous les êtres humains disparaissaient de la surface du globe, au bout de combien de temps s'éteindrait la dernière source de lumière artificielle ?

- C'est vraiment dangereux de se baigner dans une piscine pendant un orage ?

- De quelle hauteur faudrait-il laisser tomber un steak pour qu'il soit cuit en arrivant au sol ?

- Quand la bande passante d'Internet dépassera-t-elle celle de FedEx, si elle y parvient un jour ?

- Combien de tweets différents sont possibles dans notre langue ?

- Et combien de temps faudrait-il à la population mondiale pour tous les lire à haute voix ?

- Quel serait le résultat si tous les candidats au code de la route répondaient au pif au questionnaire à choix multiple ?

- Combien répondraient juste à l'ensemble des questions ?

Questions extraites du livre 'Et si ...?' de Randall Munroe.

EXERCICES

EXERCICE 1

Installer le logiciel Wireshark.
Faites quelques tests :

- Lancez une capture sur le protocole TCP, port 80
- Ouvrez un navigateur et affichez la page http://www.http2demo.io/
- Regardez ce qu'il se passe sur Wireshark
- A quoi ressemble la trame TCP ?

EXERCICE 2

Trouver quelle est la différence entre TCP et UDP.

EXERCICE 3

Avant de se lancer dans le réseau en C, découvrons **netcat**.
Lancer un serveur TCP sur le port 4242 avec **netcat.**
Dans un autre terminal, utilisez **netcat** (en temps que client TCP cette fois) pour envoyer du texte à votre serveur.

Lancer une capture Wireshark sur TCP 4242 et regarder ce qu'il se passe lors des échanges entre votre client et votre serveur.

Tips :

Man netcat

EXERCICE 4

Créer la page web suivante et utiliser **netcat** comme serveur web.
Ouvrir un navigateur et visiter http://127.0.0.1:4242
Regarder ce qu'il se passe dans Wireshark.

Terminal

```
~/> cat index.html
<html>
  <head>
     <title>Stupeflip vite</title>
  </head>
  <body>
     <h1>Moi, je suis Rascar Capac et je t'attaque avec mon Mac</h1>
     <h2>La vie une chausse-trappe, pas de quartier quand les lyrics
frappent</h2>
     <p>Mon sourire te glace comme un clic-clac qui grince</p>
  </body>
</html>
~/> printf 'HTTP/1.1 200 OK\n\n%s' "$(cat index.html)" | netcat -l 4242
```

Tips :

Il s'agit de la commande linux printf, pas de la fonction C (mais ça marche pareil les %s, etc.)
Vous pouvez servir la page à l'infini en mettant la commande dans une boucle :

Terminal

```
~/> while true; do printf 'HTTP/1.1 200 OK\n\n%s' "$(cat index.html)" |
netcat -l 4242 ; done
```

EXERCICE 5

Créer un programme client **my_tcp_client** en C qui envoie le message passé en argument au serveur.
Le serveur et le port sont passés en argument.

Le serveur sera géré avec **netcat** :

Terminal 1

```
~/> netcat -l 4242
```

Terminal 2

```
~/> ./my_tcp_client 127.0.0.1 4242 “Hello guacamole”
```

Tips :

Il y a plusieurs étapes :

- Ouvrir une **socket** (man 3 socket)
- Connecter notre socket au serveur fourni avec **connect** (man 3 connect)
- Envoyer le message avec **send** (man 3 send)

Tips 2 :

https://roscas.github.io/reseau/reseau-programmation-sockets.html

Tips 3 :

Prenez votre temps.

EXERCICE 6

Créer un programme serveur **my_tcp_server** en C qui reçoit et affiche les messages envoyés par le client **my_tcp_client** créé à l’exercice 5.

Terminal 1

```
~/> ./my_tcp_server 4242
Starting on port 4242.
Binding OK
Waiting message from client
Received => Hello guacamole
Received => How green are you?
```

Terminal 2

```
~/> ./my_tcp_client 127.0.0.1 4242 “Hello guacamole”
```

```
~/> ./my_tcp_client 127.0.0.1 4242 “How green are you?”
```

Tips :

Il y a plusieurs étapes :

- Ouvrir une **socket** (man 3 socket)
- Lier notre socket à une IP (vous pouvez utiliser INADDR_ANY pour lier à toutes les IP de votre PC) et un port avec **bind** (man 3 bind). On appelle ca “binder”
- Se mettre en écoute d’un client avec **listen** (man 3 listen)
- Attendre et accepter les connexions des clients avec **accept** (man 3 accept). Mettre dans une boucle pour recevoir plusieurs messages.

EXERCICE 7

Reprendre le programme serveur **my_tcp_server** et ajouter l’affiche de l’IP et du port des clients qui ont envoyé les messages.

Terminal 1

```
~/> ./my_tcp_server 4242
Starting on port 4242.
Binding OK
Waiting message from client
Received from 127.0.0.0:35569 => Hello guacamole
Received from 127.0.0.0:35572 => How green are you?
```

EXERCICE 8

Écrire un programme C **my_dns_resolver** qui retourne l’IP associée à une adresse DNS passée en paramètre.

Terminal 1

```
~/> ./my_dns_resolver www.skalab.fr
```

```
www.skalab.fr resolved to 188.165.53.185
```