Projet PRS

Lekë SAHATQIJA
Remy ROJAS-DELAY
3TC

Server1:

```
/*****************************
 * send
 * *************************/
                if((snd=sendto(dataSocket,imagebuff,sentsize,0,(struct sockaddr *)&dataClient_addr,clientSize))==-1) {
                        printf("send error\n");
                        exit(-1);
                }
/*****************************
* rcv
* *************************/
                 rcv=recvfrom(controlSocket,ackbuff,9,0,(struct sockaddr *)&controlClient_addr,&clientSize);
                 printf("message received: %s \n",ackbuff);
                 i++;
```

Server1 réalise la simple tâche d'émission et réception de paquets.

Bien entendu une synchronisation préalable connecte le serveur au client et accorde un nouveau port et socket “data” pour le flux d'information serveur-client.

Server2:

```
/*****************************************************************************************
Port declaration
*****************************************************************************************/
int controlPort=atoi(argv[1]);
controlServer_addr.sin_port=htons(controlPort);
controlServer_addr.sin_addr.s_addr=INADDR_ANY;

int dataPort=controlPort+1;
dataServer_addr.sin_port=htons(dataPort);
dataServer_addr.sin_addr.s_addr=INADDR_ANY;
```

Server2 fait la retransmission des paquets perdu par le client.

On a décidé d'utiliser le client2a, qui envoie les acquittements dans un port différent du celle qu'on utilise pour envoyer les données.

Server2:

```
int filesize,steps,sentsize,remaining,j,k,p;
FILE *image=fopen(filename,"r");
fseek(image,0,SEEK_END);
filesize=ftell(image);
fseek(image,0,SEEK_SET);
steps=filesize/1024 + 1;
remaining=filesize%1024;
int im=1;
/*******************************
 * TRANSMISSION STARTS
 * ******************************/
i=1;
while(i<=steps){

    if (i == steps) {
        im=fread(databuff,remaining,1,image);
        sentsize= remaining+6;

    } else {
        im=fread(databuff,1024,1,image);
        sentsize= 1030;
    }
```

On ouvre le fichier demandé par fopen() en lecture, et puis on calcule sa taille par fseek() et les variables filesize, steps et remaining.

Puis on lit le fichier, et on l'écrit sur le tableau databuff, à partir du nombre d'étapes faites.

Server2:

```
/*****************************
 * rcv stuff
 * ************************/
FD_ZERO (&fds);
FD_SET (controlSocket,&fds);

pselectControl=pselect(
        controlSocket+1,
        &fds,
        NULL,
        NULL,
        &pseudosrtt,
        NULL);

if(pselectControl==0) {
    i--;
    fseek(image,-1024,SEEK_CUR);
}else {
    rcv=recvfrom(controlSocket,ackbuff,9,0,(struct sockaddr *)&controlClient_addr,&clientSize);
}
///stop clock
if( clock_gettime(CLOCK_REALTIME,&requestEnd[0]) == -1) {
    printf("time stop error\n");
}
```

Enfin, on utilise pselect() et un timer afin de décider si un paquet a été perdu ou non. Au cas le paquet et perdu, on décrémente le compteur i et on remet le pointeur de fseek() (utilisé par fread()) à 1024 octets avant.

mainServer & serverTe2b

```
if(fork()==0){

    if(system(serverAndArgs)==-1){
        printf("error opening new server process with portNumber=%d \n",portNumber);
        exit(-1);
    }
}
```

On utilise deux fichier séparé pour le serveur multi-client.

Le premier fichier, mainServer, établie la connexion avec le client en recevant SYN, et envoyant un SYN-ACK<dataPort>.

Quand mainServer reçoit un SYN, il fait un fork() et puis, par un appel system(), lance serverTe2b avec un nouveau port (portNumber), qui est envoyé au client.

mainServer & serverTe2b

```
int filesize,steps,sentsize,remaining,j,k,p, packetDropped=0;
FILE *image=fopen(filename,"r");
fseek(image,0,SEEK_END);
filesize=ftell(image);
fseek(image,0,SEEK_SET);
steps=filesize/1024 + 1;
remaining=filesize%1024;
int im=1;
/*********************************
 * TRANSMISSION STARTS
 * ******************************/
i=1;
while(i<=steps){

    if (i == steps) {
        im=fread(databuff,remaining,1,image);
        sentsize= remaining+6;

    } else {
        im=fread(databuff,1024,1,image);
        sentsize= 1030;
    }
```

Comme le server2, serverTe2b ouvre le fichier demandé par le client, calcule sa taille et le nombre d'étapes requises pour la transmission, et puis lit le fichier par fread().