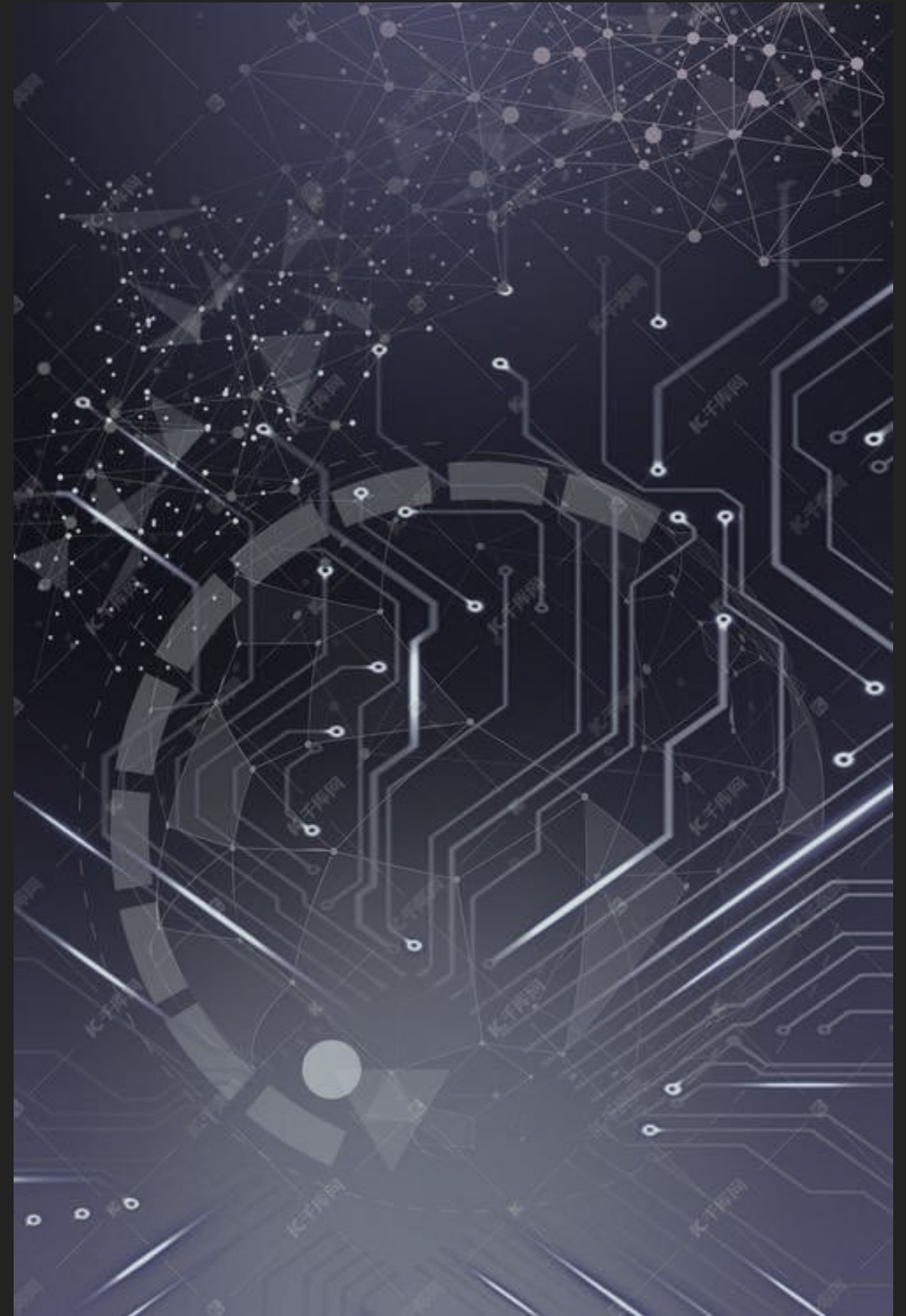


LI XINQING & SONG YUXUAN

PROJET PRS

PLAN

- ▶ Réalisation
- ▶ Performance
- ▶ Conclusion



RÉALISATION

- ▶ Structure Globale
- ▶ Three-way handshake
- ▶ Multi-Client
- ▶ Numéro de séquence
- ▶ Sliding Window
- ▶ RTT
- ▶ Congestion Control:
 - ▶ Slow Start
 - ▶ Congestion Avoidance
 - ▶ Fast Retransmit
 - ▶ Fast Recovery

STRUCTURE GLOBALE

- ▶ while(1){
 - ▶ Three Handshakes
 - ▶ process père et fils // Socket de Contrôle et Fichier
 - ▶ switch(select(..., &timeout)){ // dépend de RTT
 - ▶ case 0 //timeout
 - ▶ default //reçu un ACK
 - ▶ }
- ▶ }

THREE-WAY HANDSHAKE

- Pour le serveur:
- HANDSHAKE 1 && HANDSHAKE 2:

```
if(strcmp(recvmsg, "SYN")==0){  
    printf("Server: HANDSHAKE 1 Recieved.\n");  
    memset(sendmsg, 0, RECVSIZE);  
    strcpy(sendmsg, "SYN-ACK");  
    //Give the new port  
    char portString[5];  
    port++;  
    sprintf(portString, "%4d", port);  
    strcat(sendmsg, portString);  
    sendto(socketC, sendmsg, RECVSIZE, 0, (struct sockaddr *)&addressClient, lenClient);  
    printf("Server: HANDSHAKE 2 Sended.\n");  
}
```

handshake 1

handshake 2

variable globale

- HANDSHAKE 3

```
}else if(strcmp(recvmsg, "ACK")==0){  
    printf("Server: HANDSHAKE 3 Recieved.\n");  
  
    pid_t fpid;  
    fpid = fork();  
    if(fpid == 0){
```

handshake 3

continue

MULTI-CLIENT

▶ Process Père et Fils

▶ Père continue à suivre le boucle de contrôle

▶ Fils:

```
if(fpid == 0){  
    close(socketC);  
    //Creat Socket for Client  
    int socketU = socket(AF_INET, SOCK_DGRAM, 0);  
    if(socket < 0){  
        perror("ERROR: Cannot creat socket for Client.\n");  
        return -1;  
    }  
}
```

→ process fils

→ fermer le socket de contrôle

→ nouveau socket pour client

NUMÉRO DE SÉQUENCE

```
//Create Sequence Number
```

```
int seqRecv = 0;
```

```
int seqSend = 1;
```

sequence number reçu par serveur

sequence number envoyé par serveur

```
while((w0occupiedSize < wSizeCste) && ((len = fread(buffer, sizeof(char), RECVSIZE-6, fp)) != 0)) {  
    lenEnd = len;  
    memset(sendmsg, 0, RECVSIZE);  
    sprintf(sendmsg, "%.6d", seqSend);  
    memcpy(sendmsg+6, buffer, len);  
    printf("SEND TO CLIENT: Segment %.6d, Size: %d \n", seqSend, len);  
    sendto(socketU, sendmsg, len+6, 0, (struct sockaddr *)&addressClient, lenClient);  
    seqSend=seqSend+1;  
    w0occupiedSize = w0occupiedSize + 1;  
    memset(buffer, 0, RECVSIZE-6);  
}
```

- ▶ seqSend s'extrait de l'ACK (ACKxxxxxx)

NUMÉRO DE SÉQUENCE

- ▶ 3 relations entre seqSend et seqRecv:
 - ▶ $\text{seqRecv} == \text{seqSend} - \text{wOccupiedSize}$ Reçu le correct ACK
 - ▶ $\text{seqRecv} < \text{seqSend} - \text{wOccupiedSize}$ Perte du paquet
 - ▶ $\text{seqRecv} > \text{seqSend} - \text{wOccupiedSize}$ Retrouver le paquet précédent
- ▶ wOccupiedSize égale à la taille de fenêtre, car la fenêtre est occupée.
- ▶ seqSend est le numéro de paquet prochain



SLIDING WINDOW

```
//Create Window Size  
int wSizeCste = 1;  
int wOccupiedSize = 0;
```

la taille de fenêtre
la taille occupée dans la fenêtre

```
while((wOccupiedSize < wSizeCste) && ((len = fread(buffer, sizeof(char), RECVSIZE-6, fp)) != 0)) {  
    lenEnd = len;  
    memset(sendmsg, 0, RECVSIZE);  
    sprintf(sendmsg, "%.6d", seqSend);  
    memcpy(sendmsg+6, buffer, len);  
    printf("SEND TO CLIENT: Segment %.6d, Size: %d \n", seqSend, len);  
    sendto(socketU, sendmsg, len+6, 0, (struct sockaddr *)&addressClient, lenClient);  
    seqSend = seqSend + 1;  
    wOccupiedSize = wOccupiedSize + 1;  
    memset(buffer, 0, RECVSIZE-6);  
}
```

- ▶ $\text{seqRecv} == \text{seqSend} - \text{wOccupiedSize} \rightarrow \text{wOccupiedSize} = \text{wOccupiedSize} - 1;$
- ▶ $\text{seqRecv} > \text{seqSend} - \text{wOccupiedSize} \rightarrow \text{wOccupiedSize} = \text{seqSend} - \text{seqRecv} - 1;$

RTT

- ▶ `switch(select(maxfdp, &fds, NULL, NULL, &timeout)){...}`
 - ▶ avant le boucle: `gettimeofday(&starttime, NULL);`
 - ▶ après ACK: `gettimeofday(&endtime, NULL);`
 - ▶ `timeuse =`
$$(\text{endtime.tv_sec} - \text{starttime.tv_sec}) + (\text{endtime.tv_usec} - \text{starttime.tv_usec}) / 1000000.0$$
 - ▶ `SRTT = 3s ; SRTT = 0.8*SRTT + 0.2*timeuse;`
 - ▶ case 0: Paquet Perdu => `wOccupiedSize = 0` => Retransmit
 - ▶ default: ACK acquit.
-
- ▶ Question: Comment pouvons nous nous occuper le RTT pour le même paquet?

SLOW START && CONGESTION AVOIDANCE

▶ if(FD_ISSET(socketU,&fds))

```
if(wSizeCste>=ssthred){  
    congest = 1;  
}  
if(congest == 0){  
    wSizeCste = wSizeCste + 1;  
    wSizeFloat = wSizeCste;  
}else{  
    wSizeFloat = wSizeFloat + 1.0/wSizeCste;  
    wSizeCste = (int)wSizeFloat;  
}
```

→ limite entre les deux

slow start


congestion avoidance

▶ Timeout -> Slow Start

```
if(congest == 1){  
    ssthred = (int)wSizeCste/2;  
    congest = 0;  
}
```

FAST RETRANSMIT

► Dans le cas 0:

```
fseek(fp, -(w0occupiedSize-1)*(RCVSIZE-6)-lenEnd, SEEK_CUR);  
seqSend = seqSend - w0occupiedSize;  
w0occupiedSize = 0;  
if(congst == 1){  
    congst = 0;  
}  
wSizeCste = 1;  slow start  
wSizeFloat = wSizeCste;  
gettimeofday(&starttime, NULL);  
while((w0occupiedSize < wSizeCste) && ((len = fread(buffer, sizeof(char), RCVSIZE-6, fp)) != 0))
```

retourne au départ

FAST RECOVERY

► $\text{seqRecv} < \text{seqSend} - \text{wOccupiedSize}$:

```
//Loss of Packet
if(seqDup == seqRecv){
    nSeqDup = nSeqDup + 1;
}else{
    seqDup = seqRecv;
    nSeqDup = 0;
}
```

Compter le paquet dupliqué

```
if(nSeqDup >= 3){
    nSeqDup = 0;
```

→ Le paquet est perdu

```
printf("\n3 Duplicated ACK!!! Fast Recovery Here! \n");
```

```
ssthred = (int)wSizeCste/2;
```

```
congst = 1;
```

```
wSizeCste = ssthred + 3;
```

→ Aller directement dans l'état Congestion Avoidance

```
wSizeFloat = wSizeCste;
```

```
fseek(fp, -(wOccupiedSize-1)*(RCVSIZE-6)-lenEnd, SEEK_CUR);
```

```
seqSend = seqSend - wOccupiedSize;
```

→ retourner seqSend au paquet trompé

```
wOccupiedSize = 0;
```

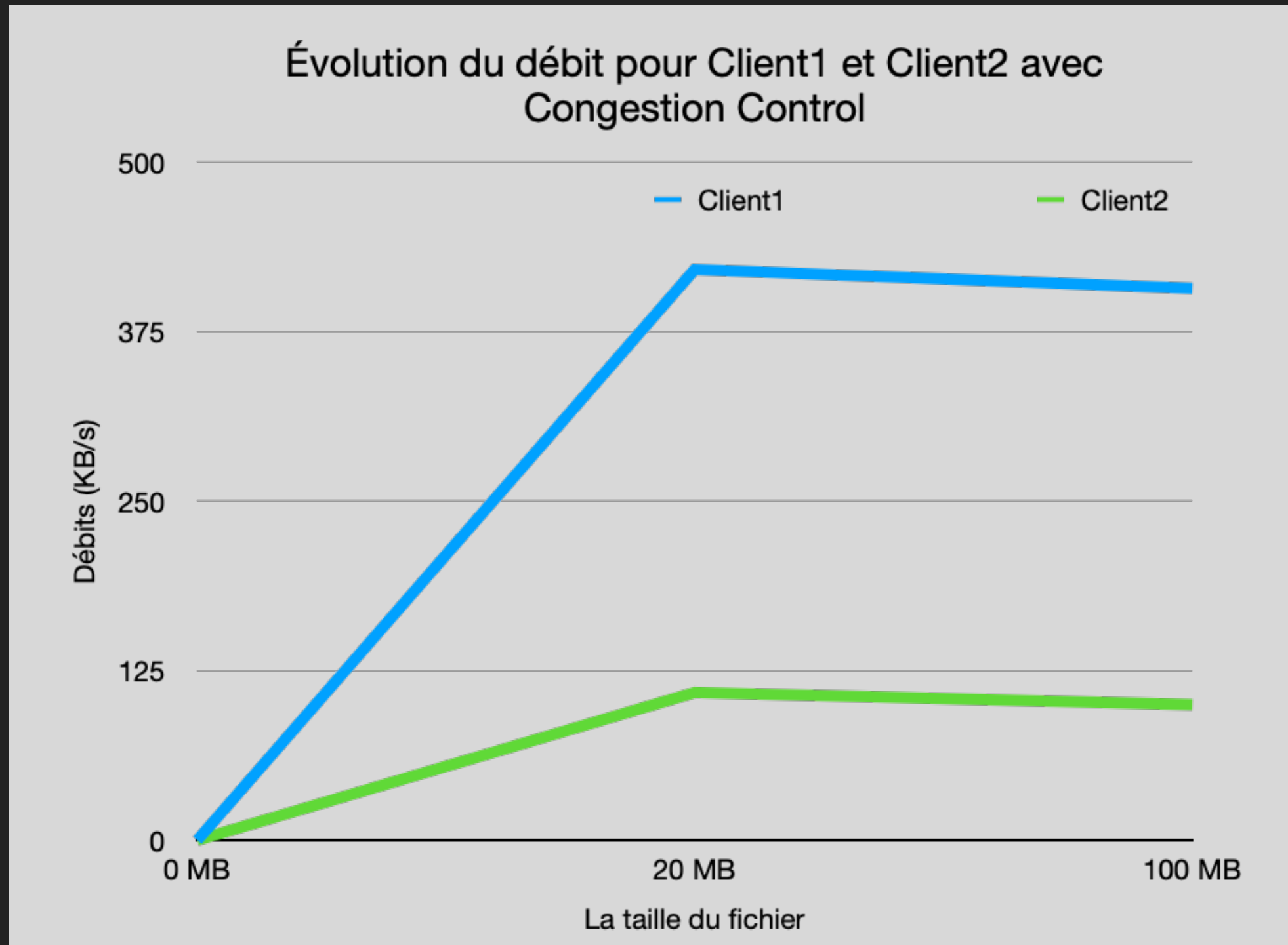
```
gettimeofday(&starttime, NULL);
```

```
while((wOccupiedSize < wSizeCste) && ((len = fread(buffer, sizeof(char), RCVSIZE-6, fp)) != 0)){
```

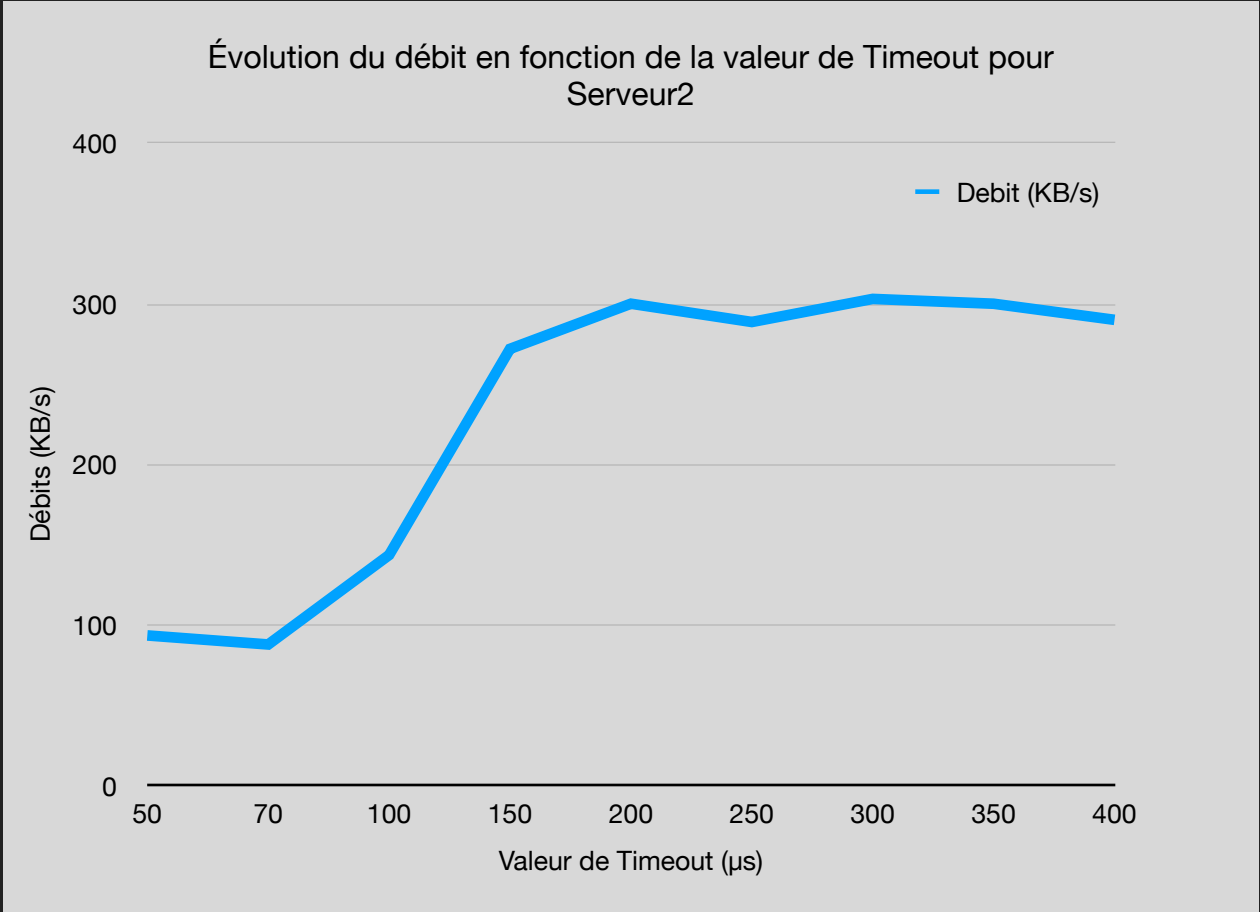
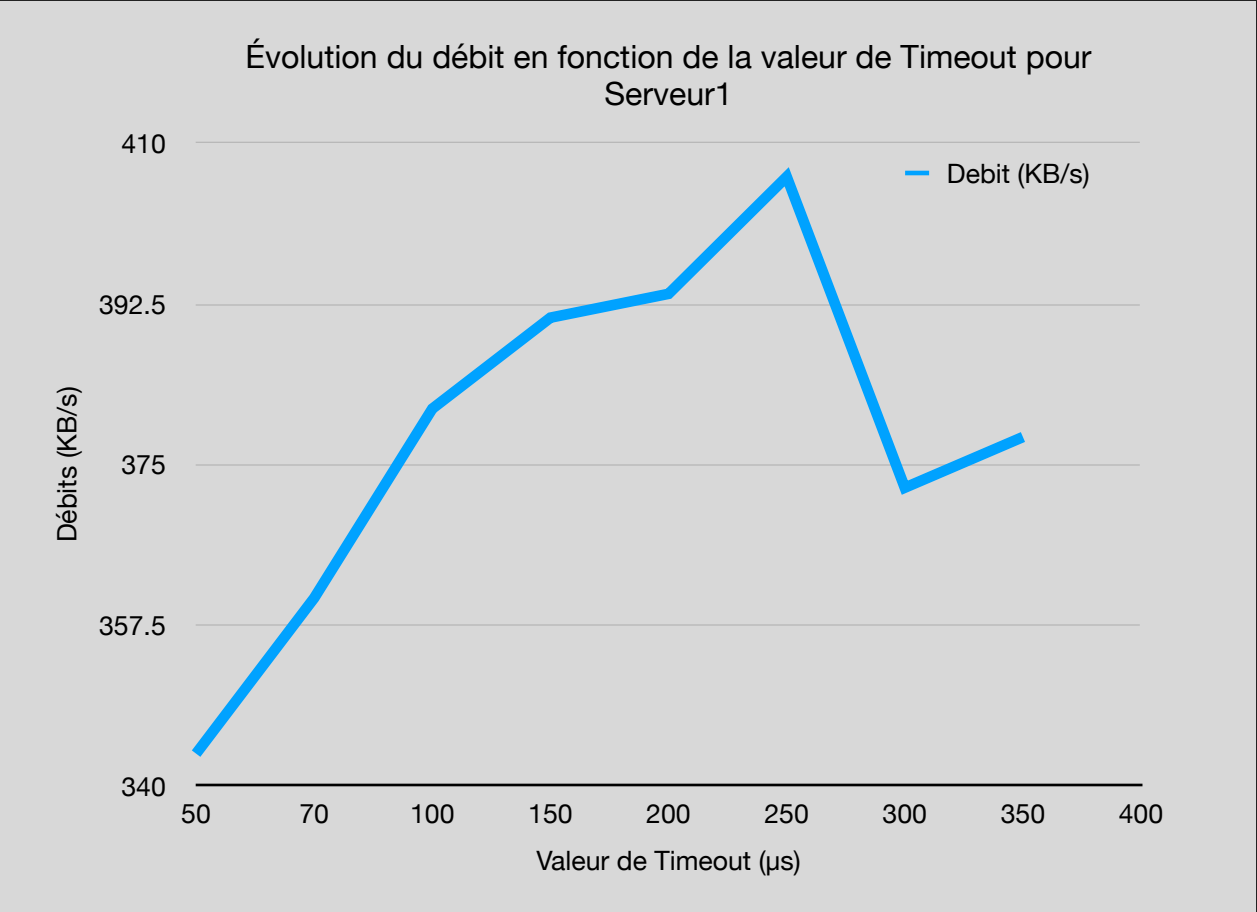
PERFORMANCE

- ▶ Performance Globale
- ▶ Performance Avec des RTT fix
- ▶ Performance Avec des fenêtres fixes

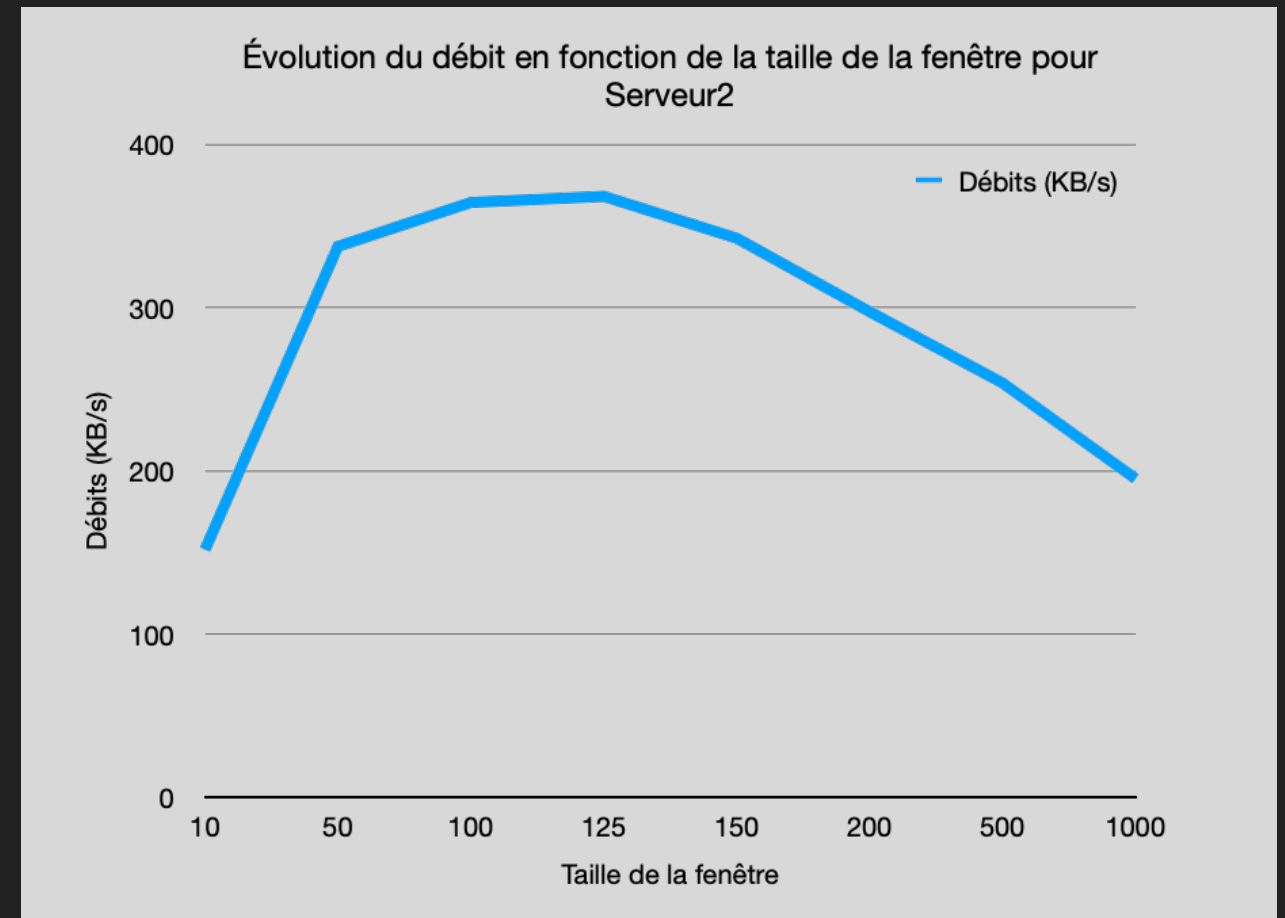
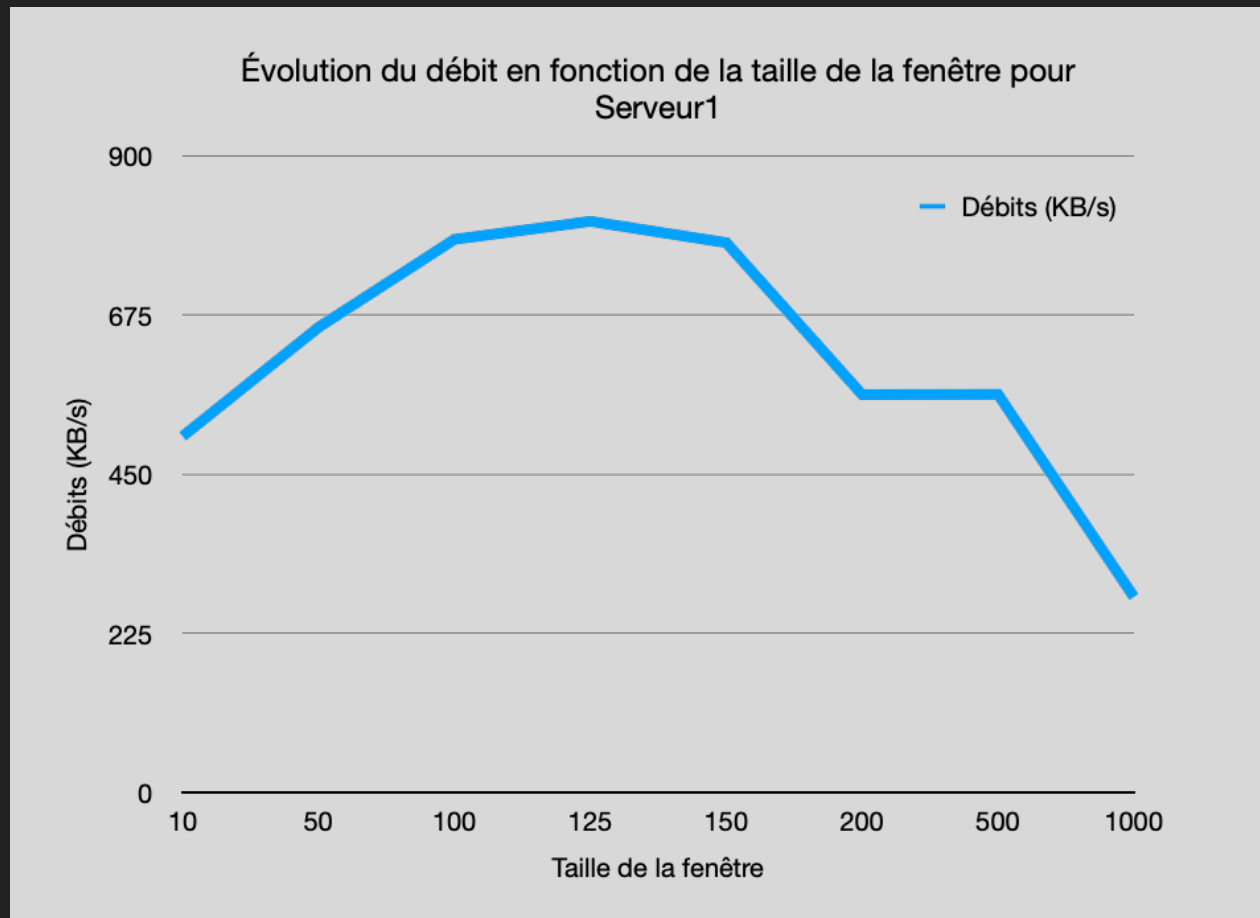
PERFORMANCE GLOBALE



PERFORMANCE RTT



PERFORMANCE DE FENÊTRE



CONCLUSION

CONCLUSION

