PROGRAMMATION Réseaux & Systèmes

Groupe : GreDel

GRENIER Raphaëlle

DELHOUME Benjamin

Scénario 1

- Conception

Scénario 1

- Implémentation
  - Langage C
  - Formule SRTT $SRTT(k) = \alpha \cdot SRTT(k-1) + (1-\alpha) \cdot RTT(k-1)$
  - Création d'un processus par client (méthode Fork())
  - Thread émission-réception (Mise en place de Mutex pour les variables partagées)
  - Vérification de la réception de plusieurs ACKs et renvoi
- Axe d'amélioration
  - Ajustement dynamique de la taille de fenêtre (exemple Slow Start)
  - Mise en place d'un seuil pour la gestion de congestion

Optimisation

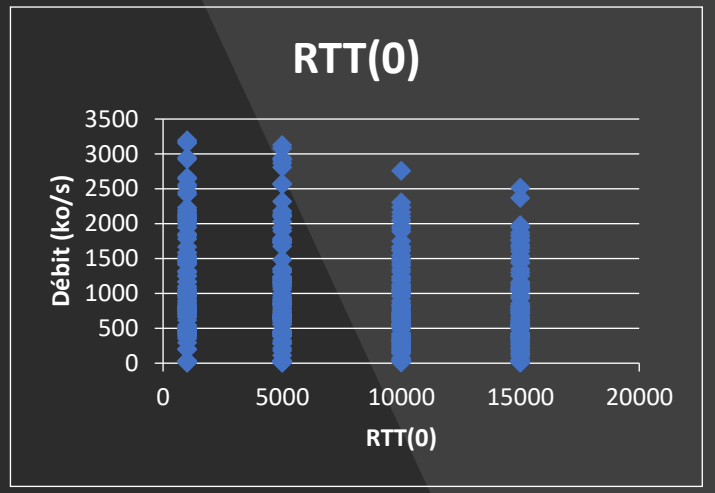
- Mise en place d'un script de test :
  - Shell
  - Résultats dans un fichier .log
  - Fichier de 1Moctets pour les tests
  - Graphe des résultats

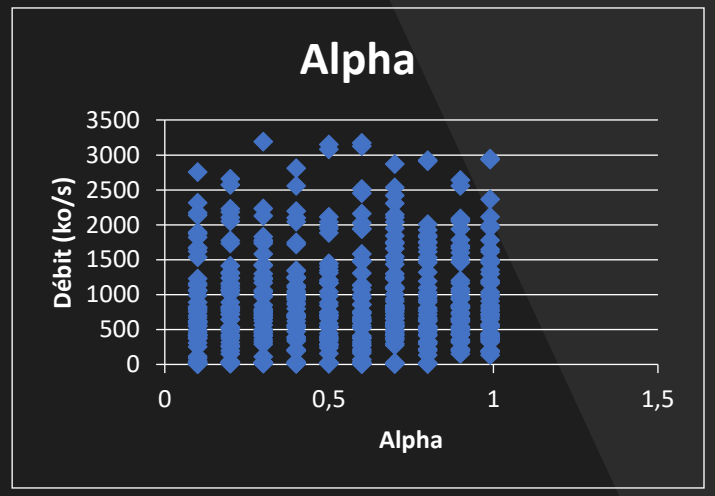
- Amélioration
  - Tests avec des fichiers plus conséquent
  - Formatage des données (Excel)

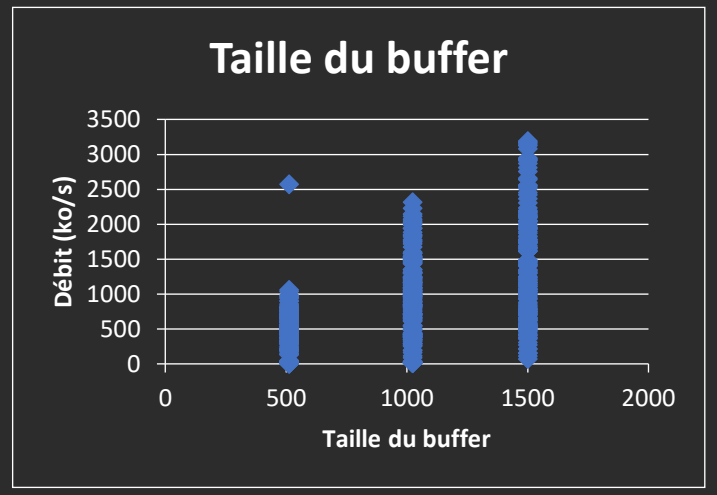
```
echo "$server_address"

for FILE_BUFFER_SIZE in '1500'
  do
    for ALPHA in '0.1' '0.2' '0.3' '0.4' '0.5' '0.6' '0.7' '0.8' '0.9' '0.99'
      do
        for TIMEOUT in '500' '1000' '5000' '10000' '15000'
          do
            echo "client $i"
            echo "FILE_BUFFER_SIZE : $FILE_BUFFER_SIZE"
            echo "ALPHA : $ALPHA"
            echo "TIMEOUT : $TIMEOUT"

            #sed -i -e "s/#define FILE_BUFFER_SIZE $ANCIENT_FILE_BUFFER_SIZE/#define FILE_BUFFER_S
            sed -i -e "s/#define ALPHA $ANCIENT_ALPHA/#define ALPHA $ALPHA/g" funcs.h
            sed -i -e "s/#define TIMEOUT $ANCIENT_TIMEOUT/#define TIMEOUT $TIMEOUT/g" funcs.h

            #ANCIENT_FILE_BUFFER_SIZE="$FILE_BUFFER_SIZE"
            ANCIENT_ALPHA="$ALPHA"
            ANCIENT_TIMEOUT="$TIMEOUT"

            make server
            ./server 1234 2000 > server.log 2>&1 &
            sleep 1
            time ./client2 $server_address 1234 1M.bin 0
            sleep 1

            killall server
            rm server.log
```

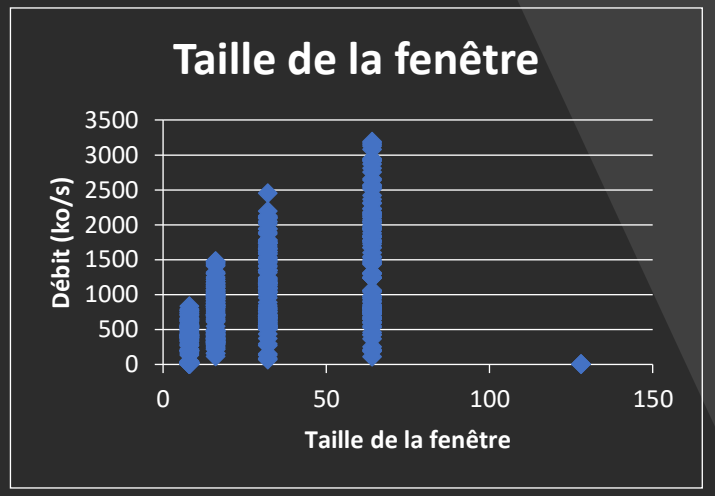
Résultats 1ère passe

- 600 Tests → 50 erreurs
  - Validation du scénario de test
  - Amélioration du code
  - Première Analyse



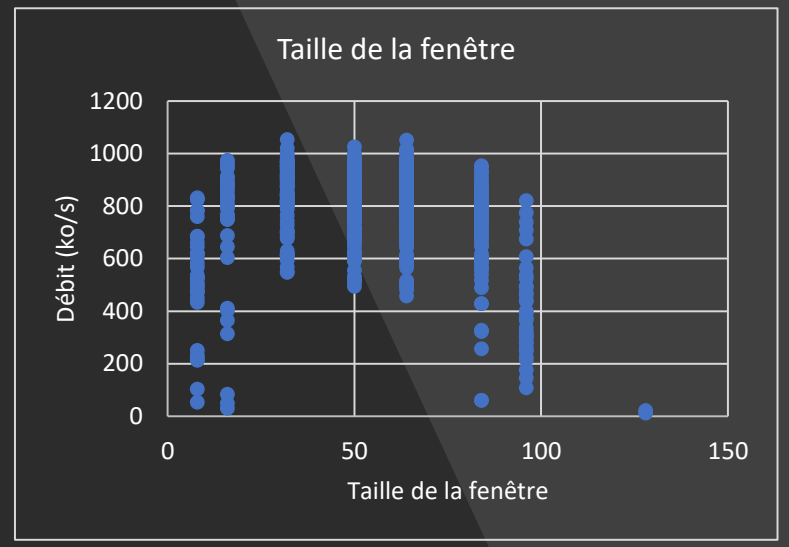




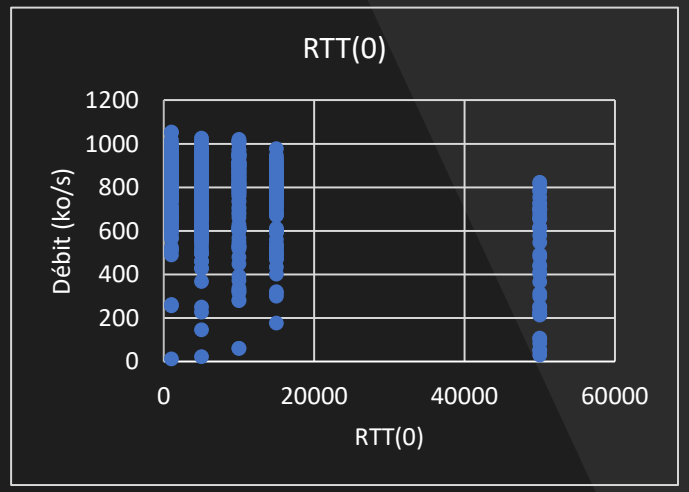


Résultats Finaux

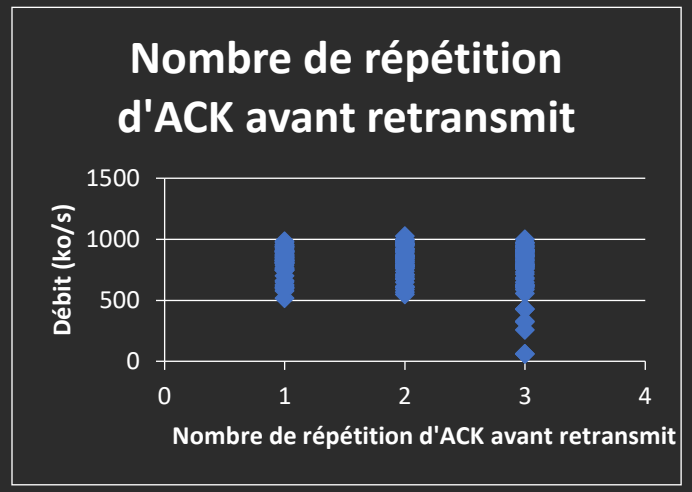
- Ajout d'une variable pour la répétition des acquittements (Fast Retransmit)
- 625 tests → 10 erreurs (Mauvais paramétrage)



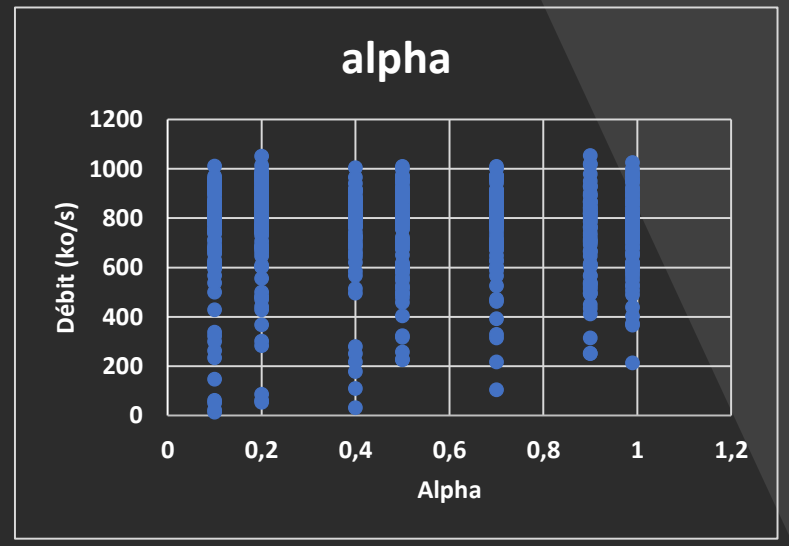
64



1000



2



0,9

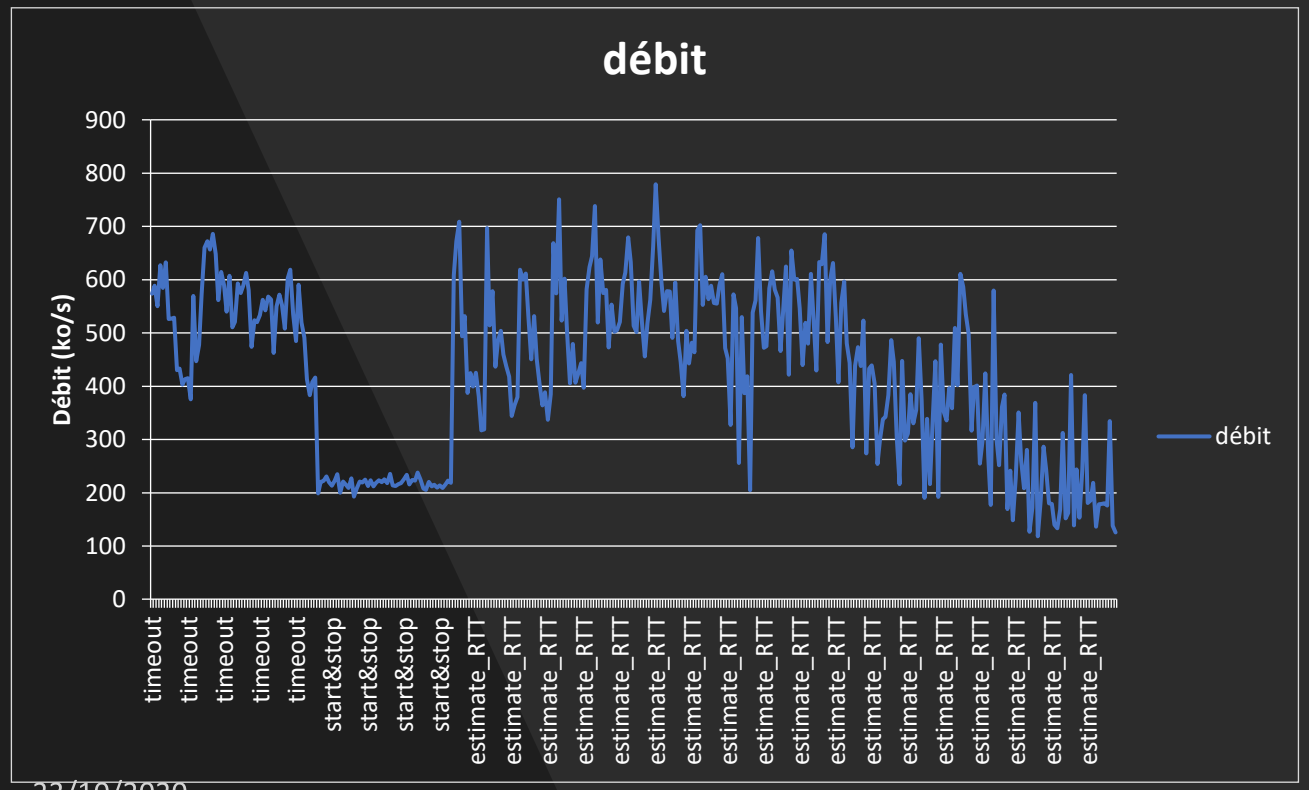
Scénario 2

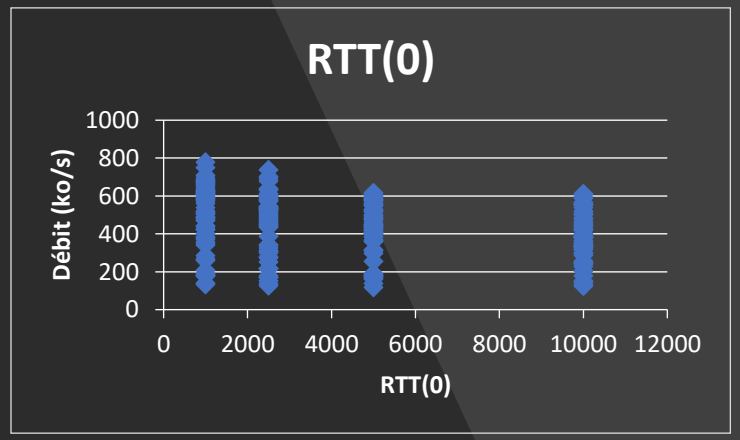
- Conception
  - Concepts identique au scénario 1
- Implémentation

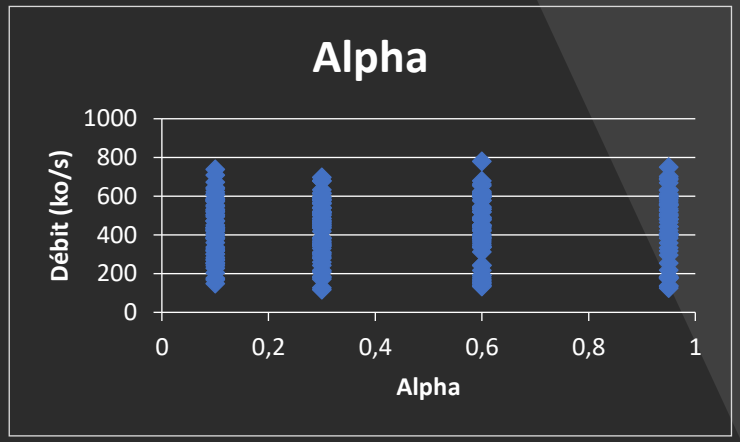
- Axe d'amélioration
  - Meilleure étude du scénario
  - Fenêtre glissante dynamique
  - Gestion de la congestion

Tests Client2

- 350 tests sans aucune erreur avec les 3 scénarios suivants:
  - Fenêtre glissante avec Timeout fixe
  - Stop & Go
  - Fenêtre glissante et SRTT







Scénario 3

- Multiple Client 1
  - Gestion du multi-clients → 1 processus crée par client
  - Lecture concurrente d'un fichier dans plusieurs processus

- Axe d'amélioration
  - Similaire au scénario 1

- Test déjà effectué pour le scénario 1

CONCLUSION

```
while (!send_done)
{
  FD_ZERO(&receive_fd_set);
  FD_SET(server_udp_data, &receive_fd_set);

  int rcv = select(FD_SETSIZE, &receive_fd_set, NULL, NULL, &timeoutvalue);
  if (rcv < 0)
  {
    error("Error during select");
  }
  else if (rcv == 0)
  {
    //printf("Timeout, Resending\n");
    pthread_mutex_lock(&mutex);
    spot = findMin(sendList);
    sequence_repeat = sendList[spot];
    if (sequence_repeat == 0)
    {
      sequence_repeat = 1;
      timeoutvalue.tv_usec = TIMEOUT;
    }
    else
    {
      receive_time = getTime();
      estimate_RTT = estimateRTT(timeList[spot], receive_time, estimate_RTT);
      timeoutvalue.tv_usec = estimate_RTT;
      //printf("Estimate RTT %ld \n", (timeoutvalue.tv_usec));
    }
    //printf("Séquence la plus petite en timeout %d\n",sequence_repeat);
    pthread_mutex_unlock(&mutex);
  }
  else
  {
    int ack_msglen = recvfrom(server_udp_data, ack_buffer, 9, 0, (struct sockaddr *)&c
```