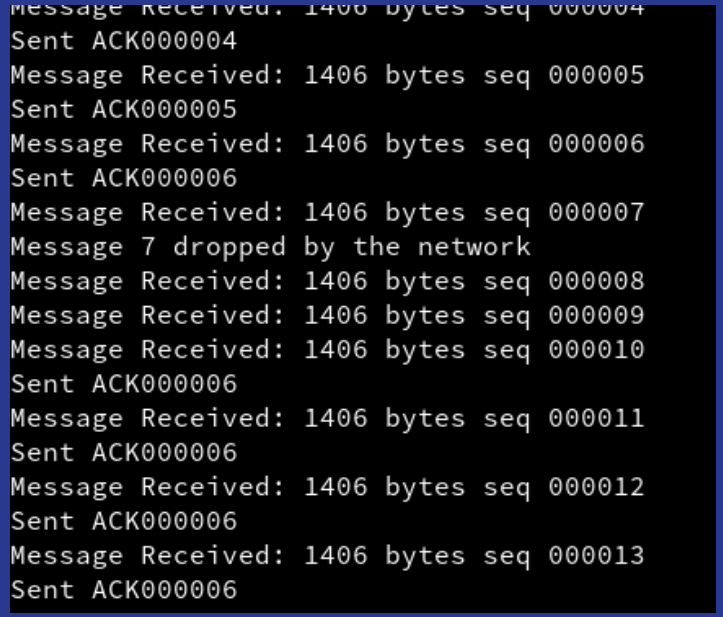
HAUT DEBIT(PRS)

NGO TUAN KIET
OUSSAMA GARAAOUI
4TC2

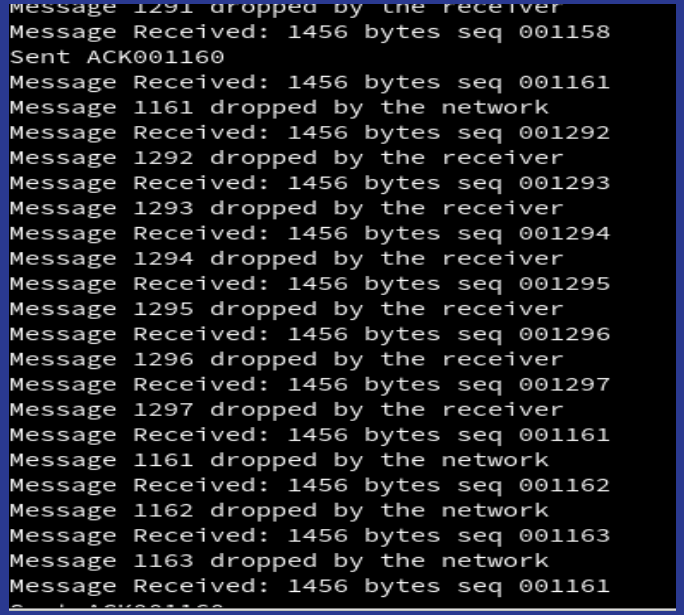
Problématique du client

Client 1

La perte du packet transmit (Scenario 1)



La perte du packet transmit (Scenario 2)



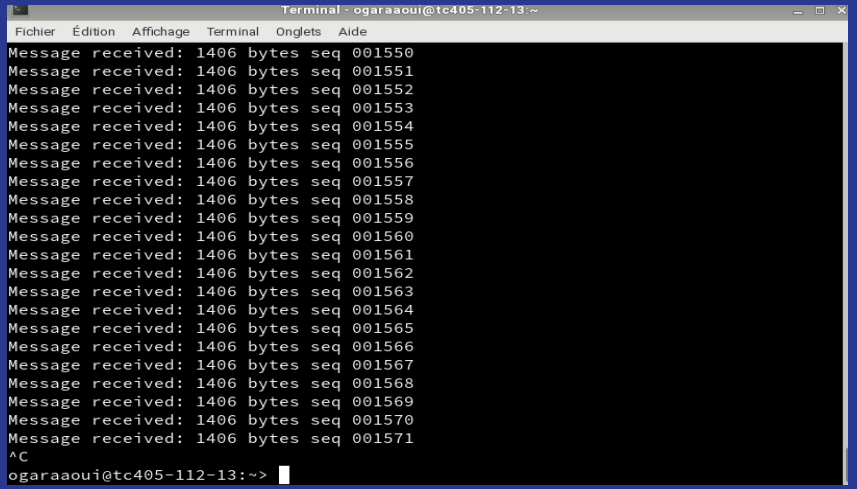
Problématique du client

Client 2

La perte du packet ACK

```
ACK000599 dropped by network
Sent ACK000599
Sent ACK000599
Sent ACK000599
Sent ACK000599
Sent ACK000599
Sent ACK000599
Sent ACK000599
```

Le temps d'envoi ACK



Paramètre constant

- Taille du buffer : 1400 octe
- Taille du numéro de séquence : 6 octe
- Timeout pour la connexion du socket principal : 40s
- Timeout pour la connexion du socket enfant : 20s
- Fichier pour le test général : 12mb.pdf

Mécanisme général

Initialisation de la connexion

Etape 1: *gère la connexion avec le client (envoi du numéro de port, SYN ACK..)*

Etape 2: *récupère le nom du fichier, le vérifie s'il existe dans le répertoire. Lis tout son contenu et les combine avec le numéro de séquence 00000K ($K \in [1,N]$) ,ensuite les stocke dans une liste de bytes*

Etape 3 *: transmet **M** élément de cette liste chaque fois. Après envoyer, le serveur écoute sur sa porte pour recevoir **M** ACKs attendus (chaque écoute ayant un timeout **λ**), mais ne sauvegarde que le plus grand ACK reçu à la fois, et ignore les ACK inférieurs aux ACK reçus précédemment*

Mécanisme pour clients

Modèle 1

- **Envoie un seul paquet chaque fois (M=1),**
- **En recevant 3 fois le même ACKs (Duplicated-Ack), renvoie le paquet perdu**
- **Estimer le RTT, puis le timeout λ à chaque envoi et l'applique la prochaine fois**

Debits ===> Client 1 : 0.17 mb/s
Client 2 : 0.06 mb/s

Mécanisme pour clients

Modèle 2

- **Slow Start (M de base = 2)**
- **Slow Start threshold = 32**
- **λ = 0.9 ms**
- **En recevant 3 fois le même ACKs, renvoie le paquet perdu, M=M/2, jusqu'à la réception du nouveau ACK, M = 2 et recommencer le Slow Start**
- **Pendant M écoutes, s'il n'y a aucun ACKs, M = 1 jusqu'à la réception du nouveau ACK, M = 2 et recommencer le Slow Start**

Debits ===> Client 1 : 0.23 mb/s ; Client 2 : 0.07 mb/s

Mécanisme pour client 1

Modèle 3.1

- **M constant au debut, M ∈ [5,20]**
- **λ constant au debut, λ = 0.3 ms**
- **En recevant 3 fois le même ACKs, renvoie le paquet perdu, M=M/2, jusqu'à la réception du nouveau ACK, si ce ACK ne correspond pas à ce que le serveur vient d'envoyer, (n°ACK < n° de dernier SEQ - M) renvoie le paquet de n°SEQ = ce n°ACK +1 et M reste à M/2, et sinon, M remet et commencer comme au début**
- **Pendant M écoutes, s'il n'y a aucun ACKs, M = 1, renvoi à partir du plus grand ACK reçu, jusqu'à la réception du nouveau ACK, M remet et commencer comme au début. Si n°ACKs < M , *λ = 1+1/(M - n°ACKs)* , M reste le même**

Debits ===> Client 1 : 0.91 mb/s

Mécanisme pour client 2

Modèle 3.2

- **M constant au debut, M ∈ [20,30]**
- **λ constant au debut, λ = 2 ms**
- **En recevant 3 fois le même ACKs, renvoie le paquet perdu, M=M/2, jusqu'à la réception du nouveau ACK**
- **Pendant M écoutes, s'il n'y a aucun ACKs, M = 1, renvoi à partir du plus grand ACK reçu, jusqu'à la réception du nouveau ACK, M remet et commencer comme au début. Si n°ACKs < M , $\lambda = \lambda + \lambda/(M - n°ACKs)$, M reste le même**
- **A la reception, si le serveur reçoit le ACK correspondant à la dernier paquet qu'il vient d'envoyer (n°ACK = n° SEQ), il passe directement à l'étape d'envoyer des paquets sans attendre les autres ACKs**

Debits ===> Client 1 = 0.84 mb/s

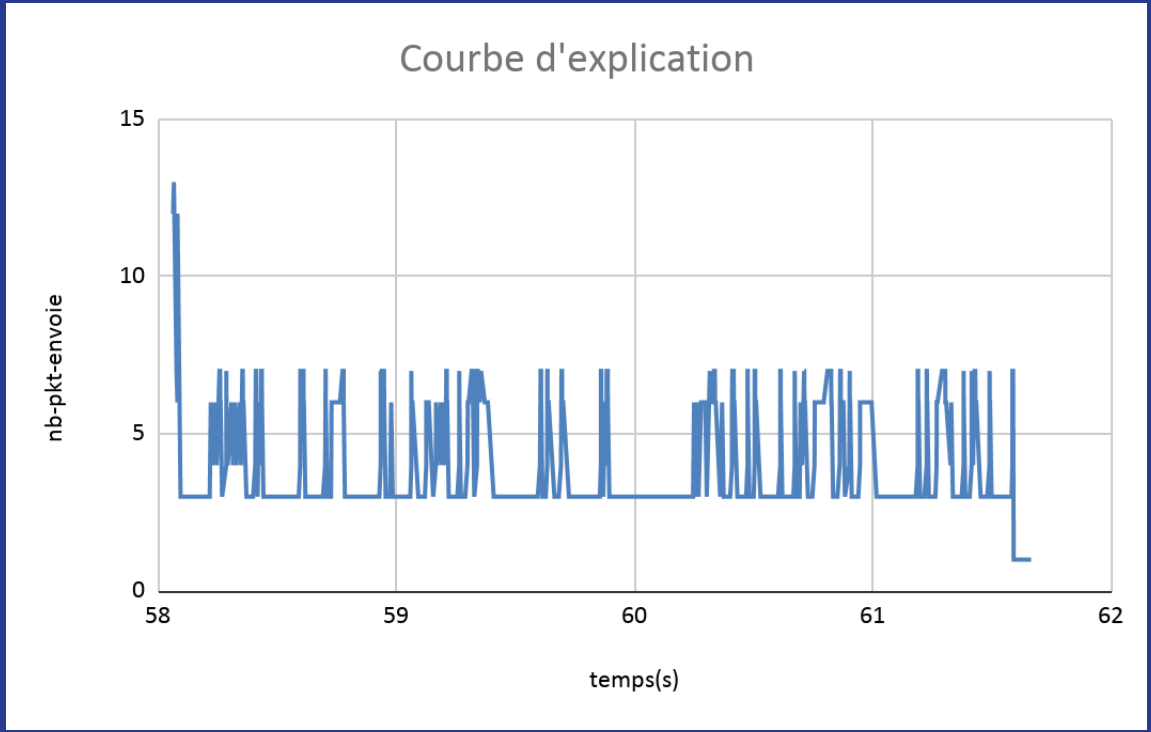
Mécanisme pour multiclients

Modèle 4

- **Utilise le module MultiThread du Python**
- **Timeout pour la connexion du socket principal : 60s**
- **Timeout pour la connexion du socket enfant : 10s**
- **Pour chaque client de type Client1, utilise le modèle 3.1 pour leur servir**

Courbe d’explication

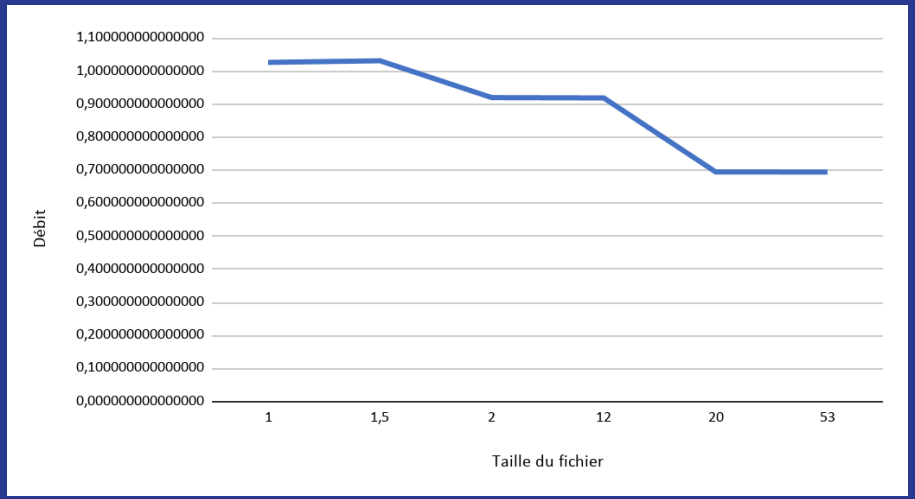
La courbe de la variation du nombre du paquet envoyé **M**



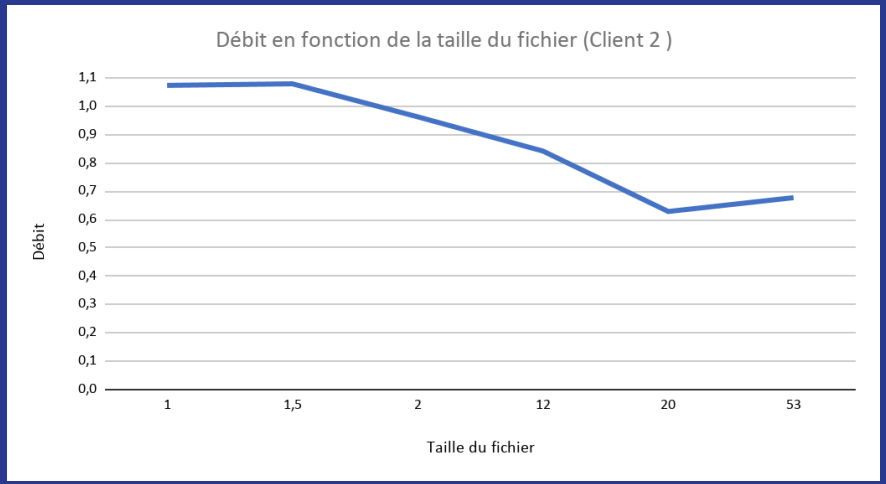
Test de performance

Débit pour des différents fichiers

Client 1



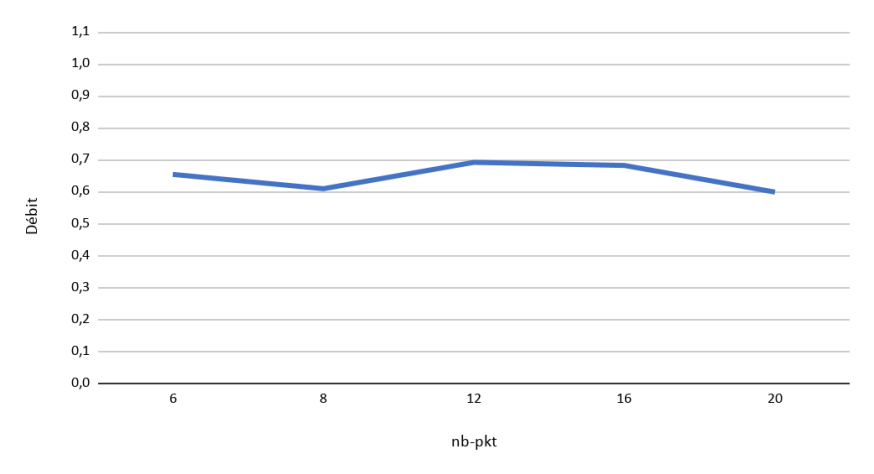
Client 2



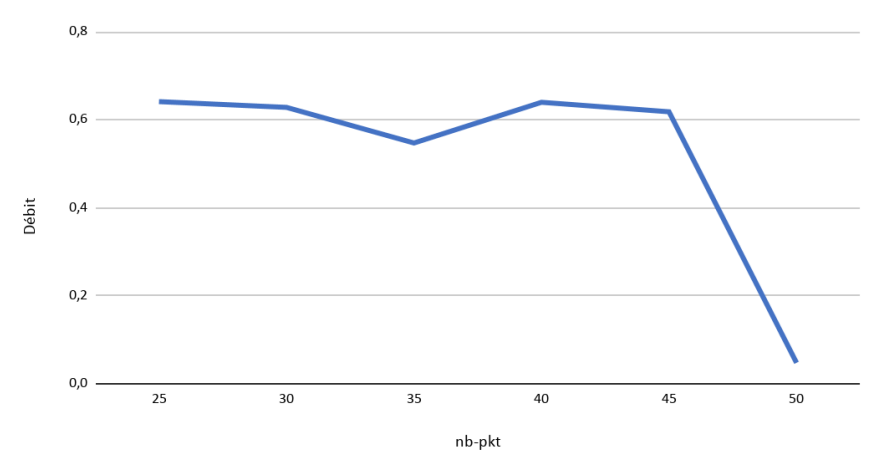
Test de performance

Débits pour différents M

Client 1



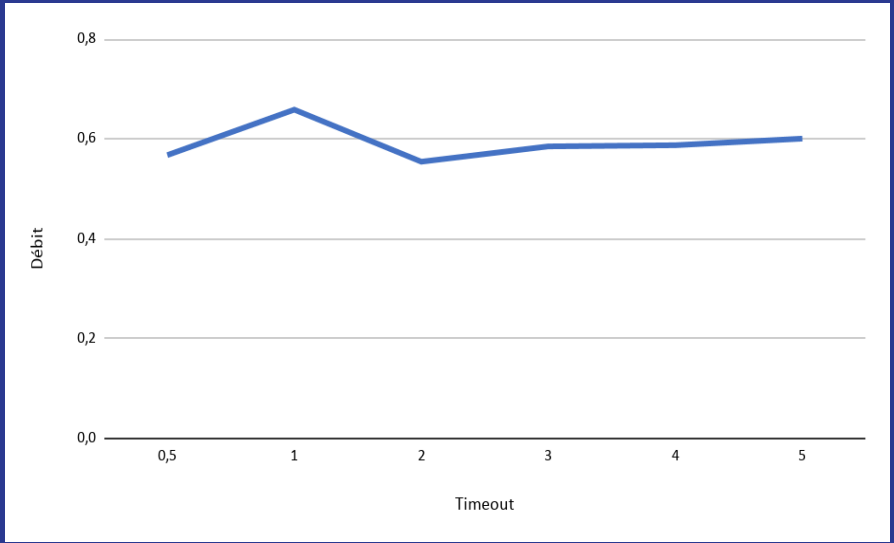
Client 2



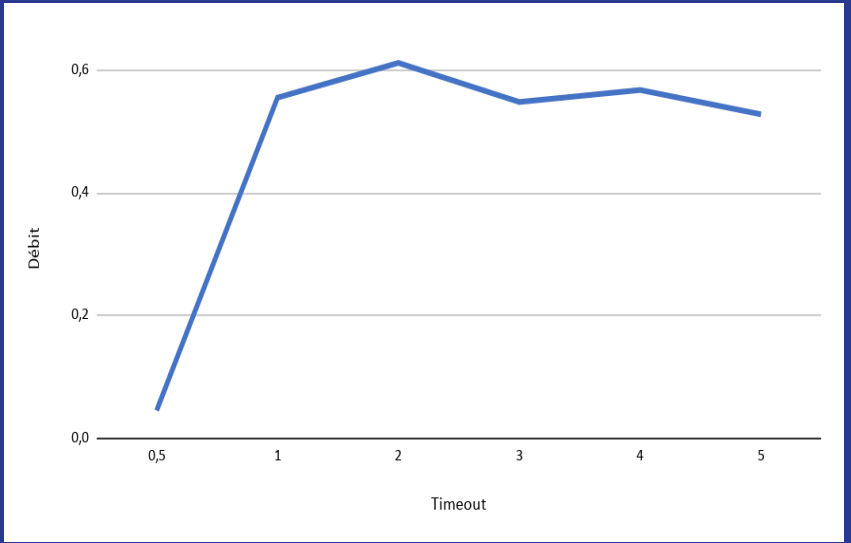
Test de performance

Débits pour différents λ (le timeout)

Client 1



Client 2



Test de performance

Débits dans des différents état du réseau

Client 1 (**M** = 12, **λ** = 3 ms)

Client 2 (**M** = 25, **λ** = 20 ms)

- Réseau perturbé :

  ⇒ Debits : Client 1 = 0.45 mb/s

  Client 2 = 0.46 mb/s

- Réseau peu perturbé :

  ⇒ Debits : Client 1 = 0.87 mb/s

  Client 2 = 0.94 mb/s

Client 1 (**M** = 16, **λ** = 3 ms)

Client 2 (**M** = 30, **λ** = 20 ms)

- Réseau perturbé :

  ⇒ Debits : Client 1 = 0.21 mb/s

  Client 2 = 0.18 mb/s

- Réseau peu perturbé :

  ⇒ Debits : Client 1 = 1.2 mb/s

  Client 2 = 0.98 mb/s

Amélioration

- Définir un algorithme d'estimation le *timeout* λ le plus dynamique
- Implémenter un structure multithread dans le serveur qui s'occupe du client 1
- Mettre en oeuvre le QoS dans le modèle de multiclient

CONCLUSION