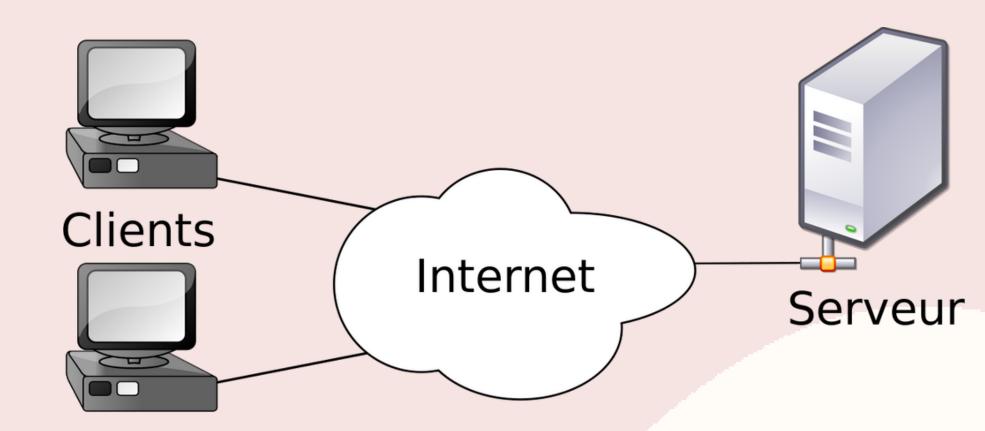
The Hunters: The best one for your Clients

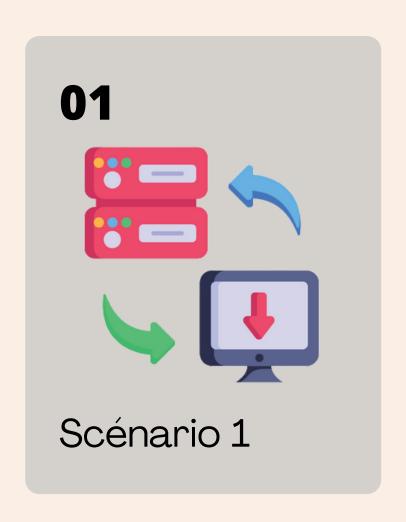
PRS: Abdellah Kabbaj Ghassen Jerbi



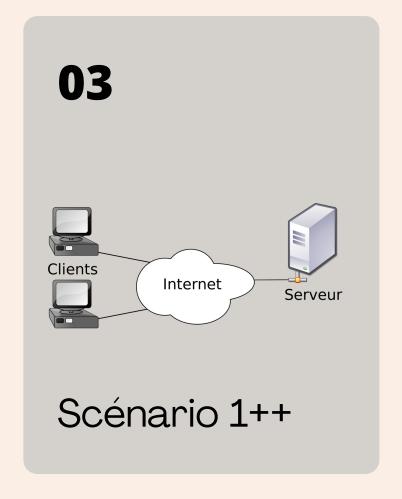
Présentation du projet

- Notre plan de projet
- Première approche
- 4 Approche Finale
- 5 Conclusion

1 Présentation du projet







2 Notre plan de projet

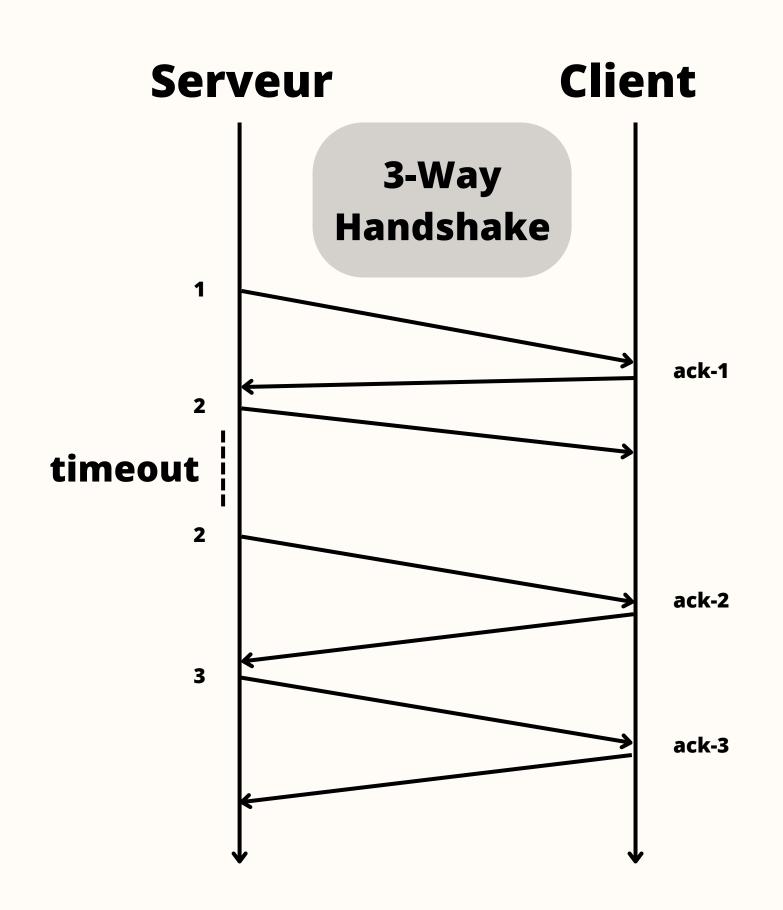
Ce qu'on prévoyait

- Fenetre glissante
- Fast recovery
- Fast Retransmit
- Thread receiving Ack

Ce qu'on a pu implémenter en plus

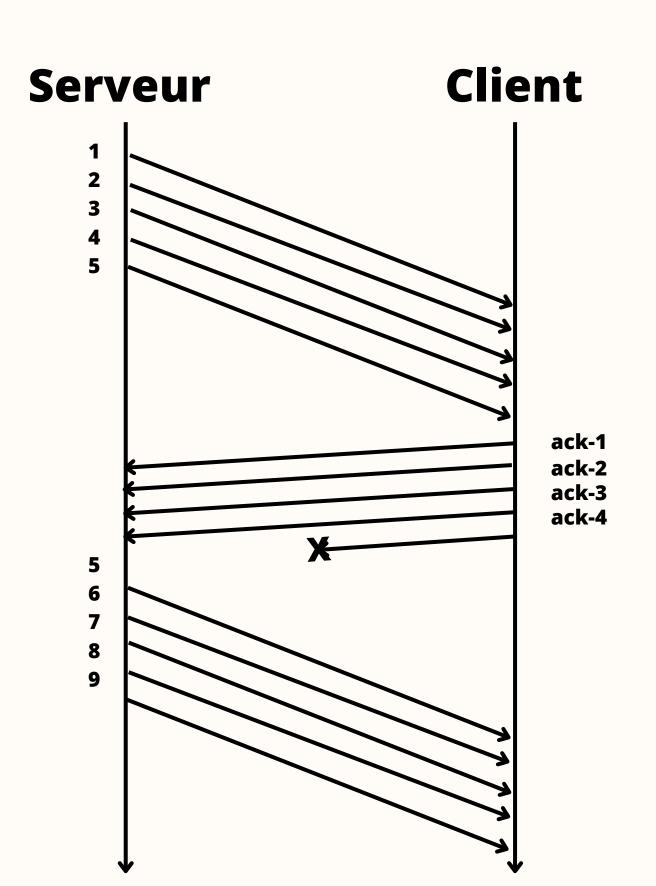
- RTT/RTO
- SlowStart
- Congestion avoidance

B Première approche



Dans un premier temps nous envoyons juste un paquet en attendant son ACK

3 Première approche

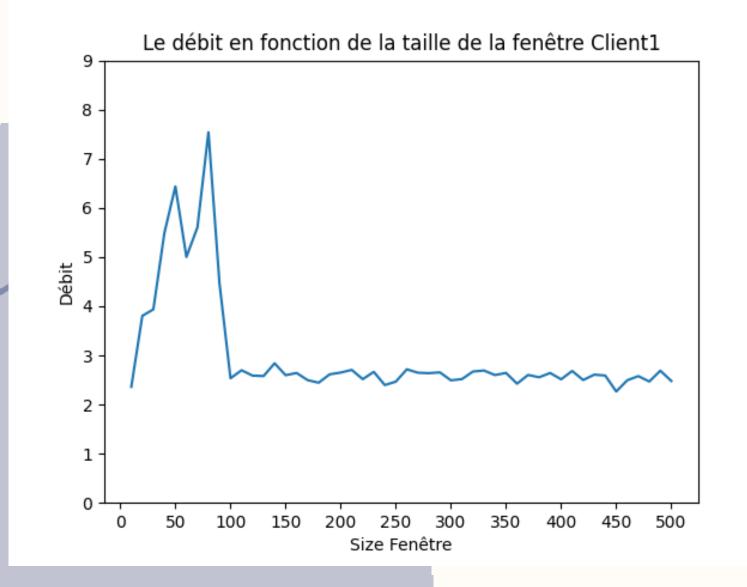


Client 1:

- Supporte une grande charge de message
- Drop de message aléatoire de 3%
- Privilégier donc un Fast Retransmit et Un Fast Recovery Rapide

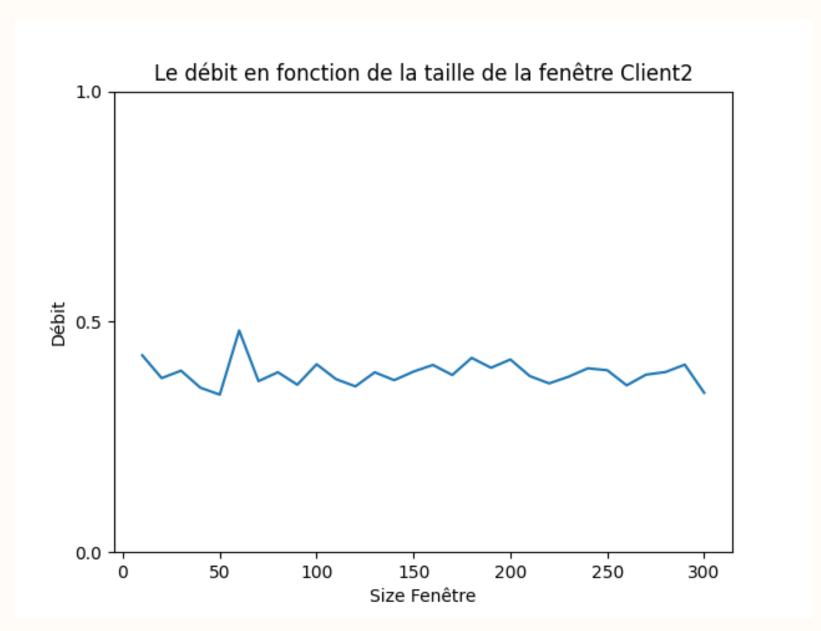
Client 2:

- Drop de message assez régulier
- Privilégier une taille de fenêtre plus petite



Évolution du débit des clients

• Taille de Fenêtre Fixe



4 Approche Finale

Utiliser C++

Nous avons décidé de passer notre projet en C++ pour pouvoir utiliser la structure <vector> pour avoir une meilleure fenêtre

Slow Start

Au vu des performances du client2 nous avons décidé d'implémenter slow start

Congestion Avoidance

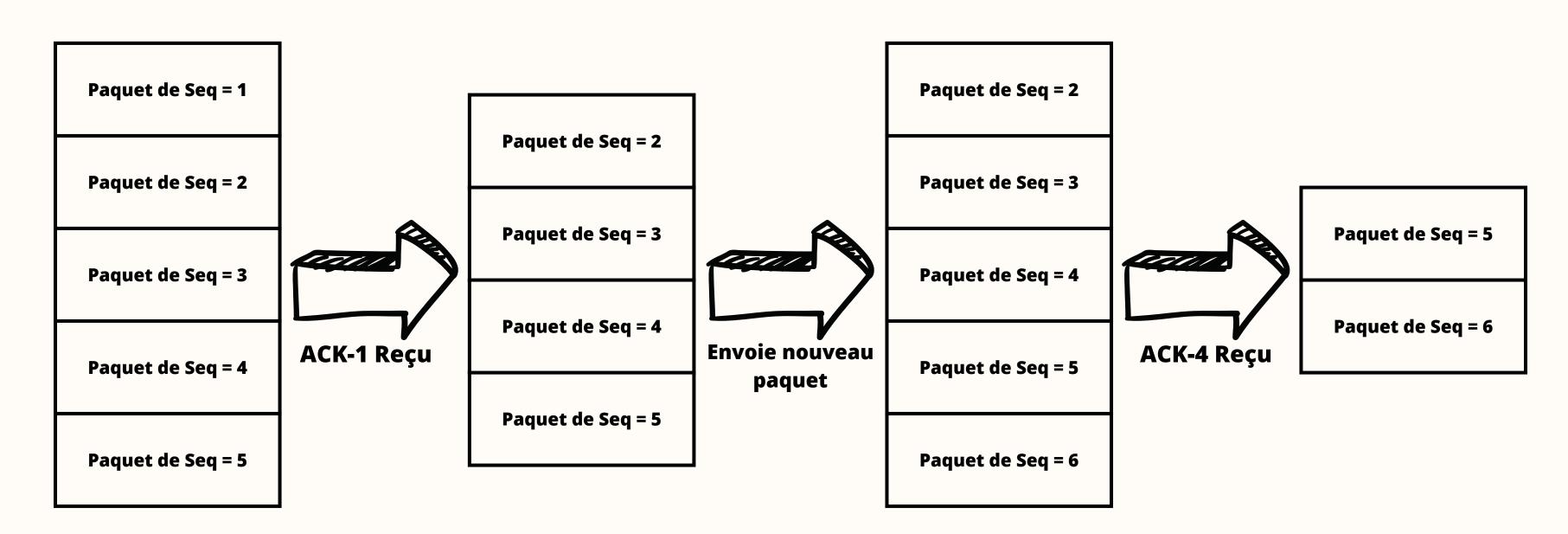
Au vu des performances du client2 nous avons décidé d'implémenter Congestion Avoidance

RTT/RTO

Pour essayer d'avoir un meilleur débit et faire en sorte que notre serveur fonctionne avec de différents clients



Approche Finale: Fonctionnement de la Fenetre:



4 Approche Finale

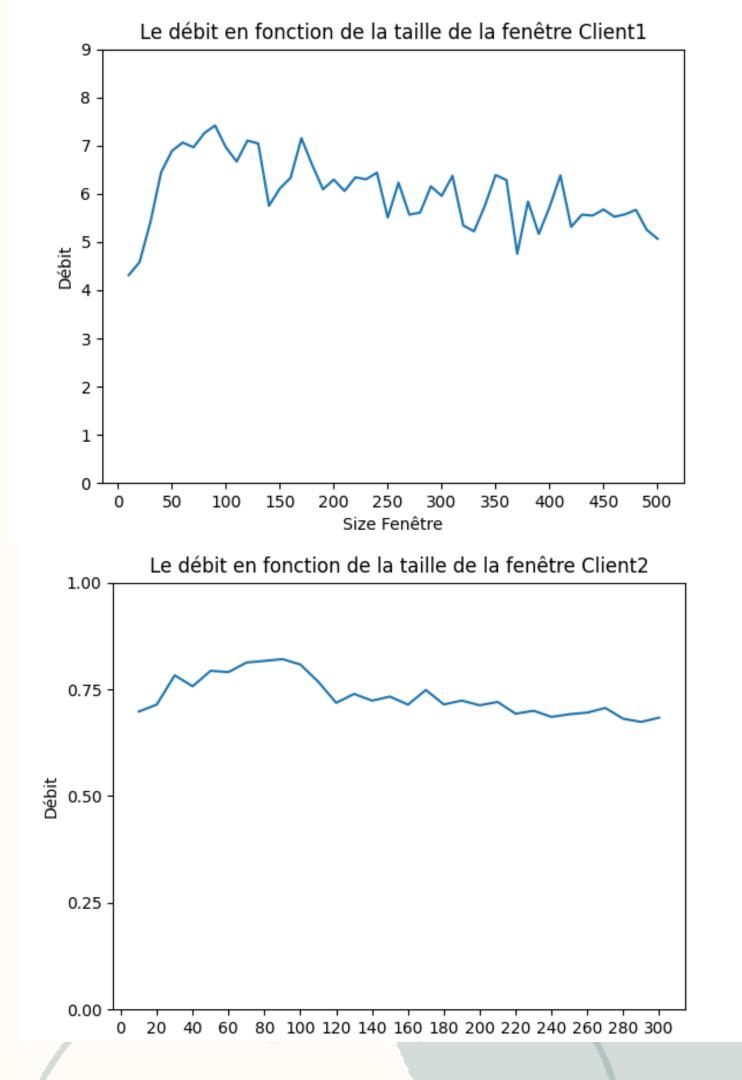
<u>Débits:</u>

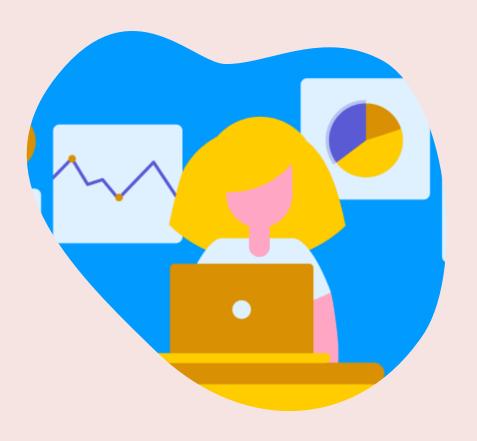
Client 1:

• 7.32 Mo/s

Client 2:

0.86 Mo/s

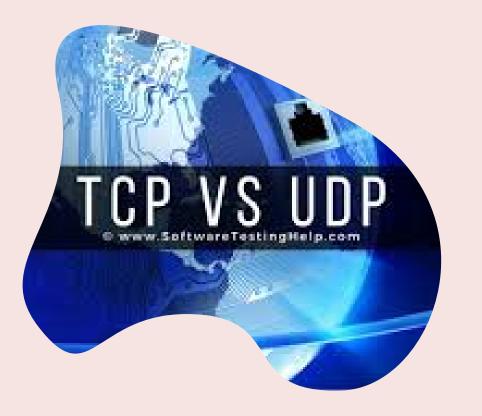




Quelques Points importants

- Thread et pas select
- Vecteurs VS Semaphores

5 Conclusion



Axes d'améliorations

- De différentes structures de données.
- De nouveaux protocoles/algorithmes
- 1 Serveur qui s'adapte à l'ensemble des clients