TP 3 - Programmation Réseau et Système

4TC PRS - INSA Lyon

Mécanismes TCP

Votre objectif pour les séances de TP restantes est d'implanter des mécanismes de contrôle de congestion classiques de la couche transport, en utilisant comme exemple les mécanismes utilisés dans des différentes versions de TCP. Plusieurs approches pourraient être envisagées dans ce sens :

- Développer vos propres modules dans le noyau Linux. L'avantage de cette solution est qu'aucune modification ne serait nécessaire dans l'utilisation de l'API Sockets : les programmes que vous avez codés pendant les deux premiers TPs seraient directement utilisables pour tester les mécanismes. Par contre, cela nécessiterait une compréhension fine du système d'exploitation et des recompilations du noyau. Heureusement pour la santé de tout le monde, ce n'est pas la solution retenue pour PRS.
- Utiliser des sockets de type SOCK_RAW, qui vous permettent d'accéder directement à la couche réseau, sans passer par une couche transport prédéfinie. Vous pourriez ainsi définir votre propre protocole transport, avec les mécanismes de votre choix. Cependant, l'utilisation des sockets SOCK_RAW nécessite des droits d'administrateur sur la machine, ce qui rend cette solution difficile à mettre en place dans les salles TP du département.
- Mettre en place des mécanismes TCP au dessus du protocole UDP. Cette solution est moins performante, car elle implique pratiquement l'encapsulation d'un nouveau protocole transport dans un autre protocole du même niveau. Malgré ces problèmes de performance, c'est bien cette approche qui sera utilisée en PRS.

La suite du sujet consiste simplement en une liste de fonctionnalités que vous devez implanter dans votre nouvelle couche transport. Vous n'êtes pas obligés de suivre l'ordre proposée dans la liste et vous êtes libres de rajouter d'autres fonctionnalités qui vous semblent intéressantes ou nécessaires. Aucune méthodologie concernant la structuration du code n'est imposée, mais le découpage en fonctions et en modules est fortement recommandé, surtout car vous serez certainement amenés à réutiliser une partie du code pour le projet.

1. Une simple communication client-serveur en utilisant UDP est un bon point de départ, surtout si vous n'avez pas intégré un client UDP dans le TP 2.
2. UDP est un protocole non-connecté, mais une couche transport qui gère le contrôle de congestion doit être connectée. Pour reproduire le comportement de TCP, avant de permettre tout échange de messages, votre client doit envoyer un segment contenant le texte "SYN". La réponse du serveur est un message "SYN-ACK", auquel le client réponds avec un message "ACK", avant d'envoyer le message utile.

3. Rappelez-vous que TCP traite les messages utiles sur une autre socket que les messages de contrôle, comme l'ouverture d'une connexion (socket créée par l'appel à la fonction *accept()*). Vous ne pouvez pas rajouter cette fonctionnalité directement au-dessus d'UDP, mais une solution est d'utiliser un deuxième port du côté serveur pour les segments de données. Le serveur devra inclure le numéro de ce nouveau port (en format chaine de caractères) dans les messages d'ouverture de connexion pour rendre la transmission de données possible du côté client .
4. Au lieu des messages texte que vous avez échangés jusqu'à ce point, transférez un fichier et vérifier qu'il est reçu correctement par le client. Utilisez des fichiers de taille moyenne (plusieurs Mb), comme des images ou des documents pdf. Vous allez devoir fragmenter ce fichier en plusieurs segments et rajouter un numéro de séquence à chaque segment.
5. N'oubliez pas que TCP acquitte chaque segment reçu. Envoyez donc un message texte "ACK_AAAAAA", ou "AAAAAA" représente le numéro de séquence acquitté, en format chaine de caractères.
6. Implanter Slow Start. Attention, pour cela vous allez avoir besoin d'une fonction qui estime le RTT.
7. Implanter Congestion Avoidance. Pour tester le comportement en état de congestion, vous pouvez forcer des pertes du coté client.
8. Fast Retransmit.
9. Fast Recovery.
10. Selective Acknowledgements.

Vous êtes fortement encouragés à chercher les spécifications détaillées des mécanismes TCP dans les RFCs correspondantes (surtout RFC 793 et RFC 5681), mais n'hésitez pas à faire des hypothèses simplificatrices quand vous considérez cela nécessaire.

Vous êtes fortement encouragés à utiliser Wireshark pour regarder le format des paquets et pour vous aider pendant le débogage.