Saulle Laura
Salamani Léandre
Robert François

Rendu TP 4 – TCP / UDP

2.1 UDP

2. Le numéro de port ne doit pas forcément être différent sur les deux machines car le socket est définie par l'identifiant de la machine et le port utilisé. Il ne peut pas y avoir de confusion. Cependant sur une même machine un port pourrait être utilisé par deux sockets différents, mais en pratique il ne faut pas car les messages se mélangeraient. Il serait donc impossible de les exploiter.

4. L'identificateur des sockets permettent d'identifier ses propres sockets. La socket étant un couple IP_port et un buffer.
La socket distante n'est pas connue. Il faut envoyer les informations sur le numéro de port de la machine distante.

5. L'entête UDP contient le numéro du port source, le numéro de port de destination, la taille du message envoyé et le CheckSum. Chaque champ étant de 2 octets.
UDP envoi à IP la taille totale qui est dans l'entête udp ainsi que le protocole utilisé (UDP ici).

6. Lorsque un message est envoyé et que le nombre d'octets lu est sup au nb d'octts envoyé, tout le message peut être récupéré.
Sinon on ne lit que la taille demandé. La suite ne pourra pas être récupéré.

Si plusieurs messages sont envoyés sans que le buffer ne puisse tout stocker, le dernier paquet reçu avant saturation sera perdu.

Lors d'un envoi sur un port inexistant, un paquet ICMP est envoyé sur le réseau.

2.2 TCP

7. La première socket est dite passive car elle reçoit seulement des demandes de connections. La seconde est active car elle fait une demande de connexion.

8. 1 socket sert juste à recevoir des demandes de connexions et 1 sert à communiquer.

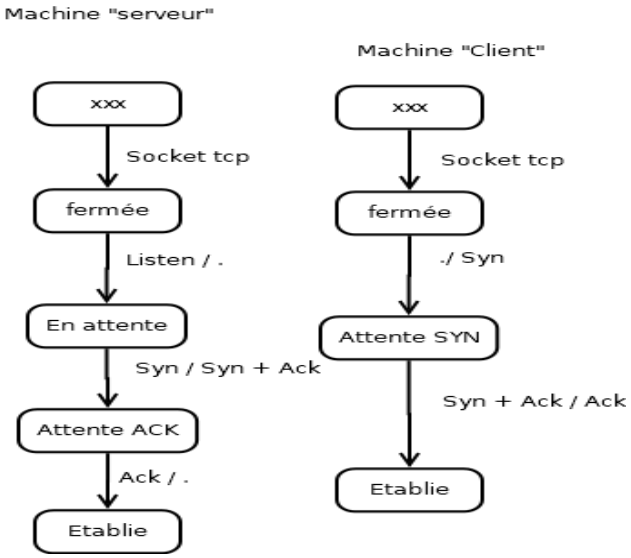
9. La station qui fait la demande de connexion envoie un paquet de type [SYN], la station a qui on a demandé la connexion l'accepte en renvoyant un paquet [SYN, ACK], enfin la station qui a envoyé la demande au départ envoie un paquet ACK pour prévenir l'autre machine qu'elle a bien reçu son acceptation, les échanges entre les deux machines peuvent alors commencer

10. Le flag SYN permet de dire à la station réceptrice que la station émettrice souhaite ouvrir une connexion. Les informations échangées durant ces étapes sont multiples : taille des fenêtres de réception (nombre d'octet que chaque station peut envoyer à l'autre avant de devoir attendre un accusé de réception), le port et l'adresse de la station qui demande la connexion pour que la station réceptrice puisse lui répondre, certaines options supplémentaires.

Les champs d'options permettent d'établir des options qui seront valable durant toute la connexion, comme la taille des fenêtre d'émissions des stations, une option permettant de calculer le RTT afin d'initialiser en fonction de cette valeur les timers de réémission et la taille maximum des segments échangés

11. accept créé une socket sur la machine serveur pour communiquer. Accept avant le connect attend, la création de la socket se fera donc immédiatement lors d'une demande de connexion.

12.



13. Une connexion est réellement identifiée par le numéro de port des deux machines.

14. états : ESTABLISHED, LISTEN (serveur), CLOSE

15. Une demande de connexion sur un port inexistant est refusée. (etat CLOSED) (RST sur wireshark)

2.2.2 Étude du séquencement et de la récupération d'erreur

16. SEQUENCE NUMBER correspond au début de la séquence envoyée
ACK NUMBER correspond au dernier octet reçu depuis le début de la communication (mal expliqué)

17. Un acquittement n'est pas toujours envoyé suite à en envoi. Il peut être envoyé suite à la réception de plusieurs séquences afin de ne pas multiplier les paquets inutiles sur le réseau.

18. Si un acquittement n'est pas reçu après un certain délai, la machine émettrice renvoi le paquet. Une fois rebranché, le paquet peut être reçu (comment?).

2.2.3 Buffer d’émission utilisé pour la récupération d'erreur

19. Si la taille u buffer est plus petite que la taille des paquets à envoyer, cela réduit le débit applicatif.

20. Le buffer d'émission permet de stoker les octets que l'on envoie à l'autre station, à partir du moment ou les octets concernés sont acquittés par l'autre machine, on peut arrêter de les stoquer dans le buffer d'émission, cependant s'ils ne sont pas acquittés avant l'expiration du timer de réémission, on utilise le fait de les avoir gardé pour les renvoyer, ou bien ne renvoyer que ceux qui n'ont pas été acquittés.

21. ?

22. ?

23. ?

24. émetteur des données :

```
SEQ = 0 , ACK = 0
Tant que connexion ouverte
        Si j'envoie des données
                SEQ = SEQ + nb_octets_envoyé
        fin si
        Si réception d'un acquittement
                Si ACK reçu < SEQ
                        réémission des octets manquant
                fin si
        fin si
fin Tant que

Récepteur des données
SEQ = 0 , ACK = 0
Tant que connexion ouverte
        Si réception de données mais fenêtre buffer plein
                envoie message spécial à l'émetteur
        Sinon
                ACK = SEQ_reçu
                (envoie d'un paquet avec le nouvel ACK )
        fin si
fin tant que
```

2.2.4 Contrôle de flux

25. l’émetteur ne peut pas envoyer plus d'octets que la taille du buffer de réception. Il pourra envoyer le reste une fois que le buffer sera libéré de la taille nécessaire.

26. Le champ window donne la taille restante dans le buffer de réception.

27. L'émetteur est débloqué lorsque l’espace disponible dans le buffer de réception du récepteur à de la place, envoie d'un paquet spécial du récepteur.

28. Le contrôle de flux se base sur le principe suivant : je peux envoyer les données si mon récepteur est capable de stoquer dans son buffer l'information, sinon j’attends qu'il me prévienne que c'est bon. Le buffer se vidant côté récepteur une fois que l'application à utilisée le contenu de celui ci.

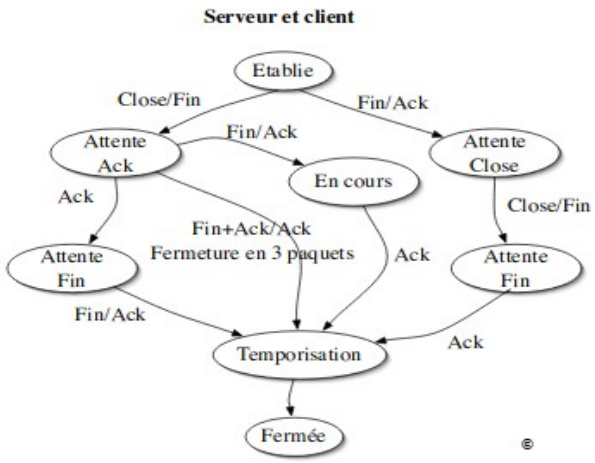
29. Non ce n'est pas intéressant car une partie de ce buffer ne sera jamais utilisé, étant donné que le buffer de réception ne pourra pas contenir autant d'informations.

2.2.5 Libération d'une connexion

30 - 31. La machine qui clos la connexion envoit [FIN,ACK], la deuxieme accepte la demande de fermeture.
Le flag FIN correspo,d au flag d'arret de connexion.
Pour terminer proprement la connexion, la deuxieme machine doit clore aussi la connexion, la première accepte aussi.

32. RST (refus) : la machine qui a clos la connexion ne peut plus emettre et donc envoit un paquet RST à l'emetteur. Celui-ci ferme la connexion, un deuxième write se soldera par un BROKEN PIPE.

33.



2.3 Exercices de synthese

34. Des premiers paquets UDP transmis (handshak pour talk) qui correspondent à une demande. Une ois la demande acceptée, une connexion TCP se créé. Un paquet est envoyé par lettre écrite. La fermeture de connexion est une fermeture basique de TCP.

35. Envoie de paquets TCP
A chaque lettre tapée, un paquet est envoyé au serveur. Celui-ci répond en renvoyant le même paquetafin de d'aquiter. Une fois la commande validée, un paquet de validation est envoyé au seveur, qui envoit le résultat de la commande.