Étude détaillée du protocole TCP La récupération d'erreur et le contrôle de flux TP N° 3

M. Heusse, P. Sicard

Introduction

L'objectif de ce TP est de comprendre les fonctionnalités du protocole TCP (Transfert Control Protocol) utilisé par la plupart des applications utilisant le réseau Internet.

Ce protocole est assez complexe vu les nombreux services qu'il rend :

- Contrôle et récupération des erreurs : TCP garantit que toutes les données envoyées seront reçues sans la moindre erreur par l'entité distante : il n'y aura ni perte, ni modification des données pendant leur transport d'une machine à l'autre.
- Ré-ordonnancement : si les données sont envoyées dans un certain ordre, elles seront rendues par TCP dans le même ordre.
- Contrôle de flux : Pour ne pas saturer le récepteur et risquer d'avoir des pertes chez le récepteur, TCP régule le flux des données en fonction de la cadence «d'absorption» par l'entité réceptrice.
- Contrôle de congestion : pour palier au problème d'embouteillage dans les routeurs intermédiaires d'Internet une régulation de l'émission est mise en place dans TCP.
- Multiplexage des accès : une ou plusieurs entités peuvent gérer simultanément plusieurs connexions TCP.

Pour réaliser ces différents services TCP est un protocole orienté connexion : cela signifie que si deux entités veulent s'échanger des données à travers TCP, elles doivent préalablement établir une connexion virtuelle. Au départ les deux entités possèdent deux rôles distincts :

- l'une est à l'écoute de demandes éventuelles, elle est dit «passive», c'est ce que l'on appelle le serveur ;
- l'autre est active, elle fait la demande de connexion. C'est le client.

Une fois la connexion ouverte, elle est symétrique et bi-directionnelle.

Le service proposé par TCP possède aussi la caractéristique d'être de type *byte-stream* (flux d'octets), il n'y a pas de découpage fixé par TCP dans le flux de données véhiculées.

La notion de port a été introduite pour permettre le multi-accés à TCP. Un port est un entier de 16 bits qui sert à identifier un point d'accès aux protocoles de la couche 4. Ainsi une connexion TCP est identifiée de façon unique par *deux* couples [adresse Internet, numéro de

port].

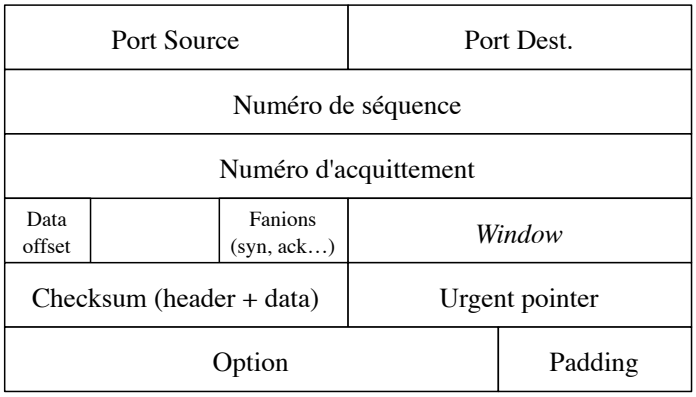

Fig. 1 – Entête TCP

L'entête TCP est représenté en figure **??**. Le champ de bits *fanions* est composé des 6 fanions suivants : URG, ACK, PSH, RST, SYN, FIN.

Signification des différents champs :

- *Source port* et *destination port* : déterminent les points d'accés à TCP par les applications de part et d'autre de la connexion.
- *Sequence number* et ACK *number* sont utilisés pour le séquencement et la récupération d'erreurs de données (le fanion ACK indique si le champ ACK *number* contient une valeur valide).
- *Data offset* indique la taille de l'entête TCP en mots de 4 octets (la taille de l'entête TCP est variable : elle peut être complétée par une ou plusieurs options de 4 octets chacune).
- *Checksum* a le même sens que dans les paquets UDP.
- *Urgent pointer* est utilisé pour le transport de «données urgentes» (le flag URG indique si le champ URGENT POINTER contient une valeur valide). Il indique les octets à traiter en priorité.
- Les flags SYN et FIN sont utilisés pour l'établissement et la fermeture des connexions virtuelles.
- Le flag RST est utilisé pour fermer les connexions virtuelles qui sont dans un état incertain (SYN dupliqués ou panne).
- Le flag PSH permet de signaler au récepteur qu'il faut délivrer immédiatement à l'application les données.
- Le champ *Window* est utilisé pour le contrôle de flux. Il contient le nombre d'octet que le récepteur peut encore stocker à partir du numéro d'acquittement contenu dans le même paquet.

Utilitaires

Pour étudier le protocole TCP, nous allons utiliser différents utilitaires :

- `socklab` : Logiciel "maison" vous proposant une interface souple et simplifiée sur les sockets. Ce «laboratoire» à *socket* vous permet en fait d'appeler de façon interactive les pri-

mitives de base de manipulation des sockets.

- `tcpmt, tcptarget` : Logiciels client/serveur vous permettant de générer des flux continus au dessus de TCP (dans le répertoire `/root/ ipmt-tools`).
- `ethereal` : Outil de capture de paquets sur le réseau. On utilisera en particulier le menu `statistics` qui permet d'obtenir des courbes intéressantes sur le comportements de TCP (évolution des numéros de séquences, évolution du RTT...)
- `dummynet` : option du noyau système permettant de modifier le comportement d'une machine au niveau du protocole IP (délai, pertes, ...).
- `sysctl` : outil de gestion des paramètres du système et en particulier des paramètres de TCP (taille des buffers, durée de timer ...)

Pour les expériences qui suivent, il sera souvent judicieux de résumer les échanges de paquets sur un croquis temporel en faisant apparaître les champs pertinents.

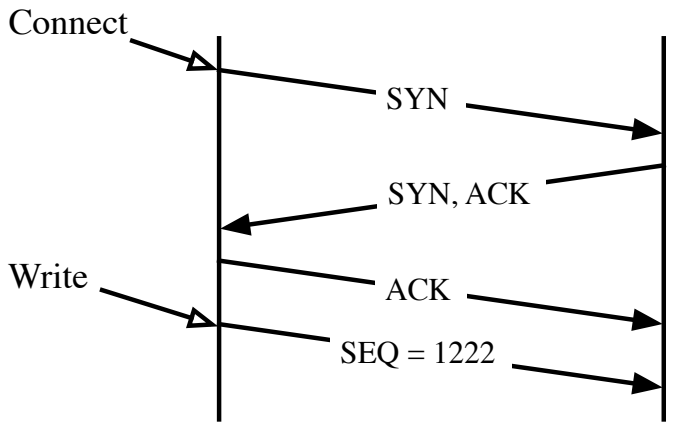

Fig. 2 – Ébauche de croquis temporel

Déroulement du TP

- Montez le réseau de la figure **??**. On pourra utiliser des câbles croisés ou des hubs à 10 Mégabit/s. En cas d'interface à 100 Mégabit/s on forcera le débit à 10 mégabit/s (`ifconfig xl0 media 10baseT/UTP`). **ATTENTION** certaines cartes n'acceptent pas le forçage manuel à 10 mégabits/s, dans ce cas on utilisera des hubs 10 mégabits/s.
- Configurez les interfaces en jeu.
- Remplissez les tables de routages afin que les 3 machines communiquent. Forcez si nécessaire la machine M2 à être un routeur (`sysctl net.inet.ip.forwarding=1`).
- Remplissez le fichier `/etc/hosts` à l'aide de noms adéquats pour les 3 machines.

1 Établissement d'une connexion

Sur les deux machines M1 et M2, lancez `socklab`.

- Capturez et analysez les paquets générés lors de l'établissement d'une connexion TCP. Décomposez les étapes de cette connexion : enchaînement dans le temps des demandes

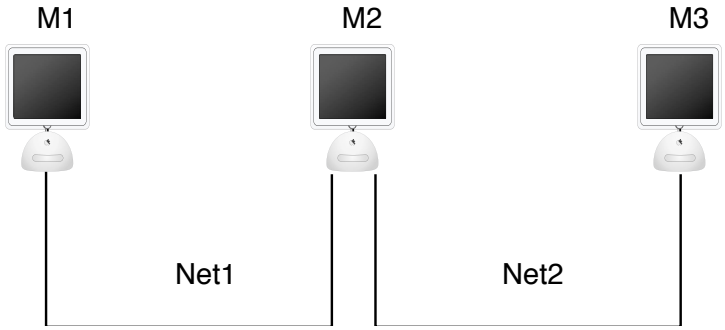

Fig. 3 – Réseau à configurer

de services à TCP et des messages échangés.

- **TCP utilise des options, expliquez à quoi sert chacune d'elle ?** :
- Ouvrez plusieurs connexions d'une machine vers un même port destinataire.

  Qu'est-ce qui identifie réellement une connexion, c'est-à-dire, comment TCP associe-t-il les messages reçus aux différentes connexions en cours ?

2 Etude de la récupération d'erreur

2.1 Numéro de séquence et d'acquittement

Dans la précédente connexion TCP, envoyez un paquets de 5000 octets de données entre les deux machines grâce à la commande `write` (à la place d'un message normal, vous pouvez utiliser la notation #nnn pour envoyer un message de *nnn* octets).

Analysez les paquets engendrés par le transport des données.

Résumez sur un croquis temporel l'expérience en faisant apparaître les champs *sequence number* et *ack. number* de l'entête TCP. Rappelez la fonctionnalité et l'utilisation de ces deux champs.Y a-t-il un acquittement par paquet de données ? Quel intérêt ?

On pourra s'aider du menu `statistics/flow graph` d'Ethereal.

2.2 Temporisateur de retransmission

Après l'ouverture d'une connexion TCP sur M1 et M3, envoyez des données vers M3 que vous aurez au préalable débranchée du réseau.

Observez ce qu'il se passe en capturant les paquets côté émetteur.

Comment évolue le Temporisateur de retransmission ? Quel intérêt ? Pourquoi observe-t-on des séries de paquets ARP au bout d'un certain temps ? Les durées des timers de ARP peuvent être observées par la commande `netstat -rn`, attention ils peuvent être assez long (vingtaine de minutes).

2.3 Temporisateur de retransmission

Exemple de configuration à l'aide de `Dummynet` :

1. Si le programme `ipfw` ne fonctionne pas, l'activer par modification du noyau : `kldload dummynet.ko`.
2. On peut avoir la liste des règles en cours par `ipfw list`
3. Rajoutez si nécessaire la règle : `ipfw add 5000 permit ip from any to any`
4. Règle de configuration d'un délai supplémentaire de 15 millisecondes pour les paquets portant l'adresse source 192.168.10.119 et l'adresse destination 10.0.0.2 :
   - `ipfw pipe 1 config delay 15ms`
   - `ipfw add 4001 pipe 1 ip from 192.168.10.119 to 10.0.0.2 in`
5. Suppression d'une règle : `ipfw delete 4001`
6. Attention les règles sont "testées" suivant l'ordre croissant de leur numéro.

Modifiez le délai "IP" sur M2 à 5ms, 10ms et 100ms et vérifiez à l'aide de `ping` ce délai.

Attention la granularité du délai n'est pas forcément de la finesse désirée.

2.4 Paramètres TCP

TCP possède de nombreux paramètres que l'on peut modifier à l'aide de la commande système `sysctl`. Regardez la liste des paramètres TCP par `sysctl net.inet.tcp`.

A quoi correspondent les paramètres `net.inet.tcp.sendspace` et `net.inet.tcp.recvspace` ?

Nous allons pour les prochaines expériences modifier certains de ces paramètres. Sur M1 et M3, effectuez les commandes suivantes :

- `sysctl net.inet.tcp.delayed_ack=0` , permet d'activer ou non le délai d'attente avant émission d'acquitement. Vérifiez sa valeur actuelle sur les captures précédemment faites (temps donnés par Ethereal).
- `sysctl net.inet.tcp.hostcache.expire=0` , permet d'inhiber le mécanisme de cache concernant les caractéristiques des connexions TCP en cours.
- `sysctl net.inet.tcp.sack.enable=0` , permet de désactiver le mécanisme d'acquittement sélectif possible dans TCP.
- `sysctl net.inet.tcp.inflight.enable=0`

2.5 Fenêtre à anticipation

- Sur la machine routeur M2 ajouter un délai de 100 ms à la traversée de la couche IP ;

- Sur la machine M1 modifiez la taille du buffer d'émission à 10000 octets.
  `sysctl net.inet.tcp.sendspace=10000`
- Ouvrez une nouvelle connexion TCP entre les machines M1 et M3.
- Envoyez un paquet de 40000 octets depuis M1 à travers une connexion TCP ouverte entre M1 et M3.
- Capturez (plutôt sur l'émetteur) les paquets circulant à ce moment là et observez l'évolution des numéros de séquences sur la courbe donnée par `Ethereal`.

- **Comprendre et analyser précisément cette courbe. On regardera aussi attentivement les acquittements émis par le récepteur.**
- **Rappeler le principe de la fenêtre d'anticipation pour expliquer cette courbe.**
- **Où retrouve t-on l'influence de la taille du buffer d'émission dans cette courbe ?.**
- **Quel autre mécanisme est mis en jeu lors de l'envoi des premiers paquets de données ?**

Utilisation de `Ethereal`

- Menu `statistics/ TCP stream graph / time sequence graph (tcptrace)`.
- Il faut sélectionner un paquet de donnée (et non un acquittement) avant d'afficher la courbe.
- En gras sont donnés l'évolution des numéros de séquence, en clair sont donnés les numéros d'acquittement reçu et les numéros d'acquittement augmentés de la valeur du champ `WIN`.
- Il est possible de zoomer sur une partie de la courbe en sélectionnant un paquet (sur la courbe) et à l'aide des 2 boutons de la souris (ensemble).
- On peut faire un zoom arrière de la même manière à l'aide de la touche majuscule.
- On peut se déplacer dans la fenêtre de la courbe à l'aide du bouton droit de la souris (maintient enfoncé plus déplacement).

2.6 Cas de pertes dans la fenêtre à anticipation

Sur la machine M2 configurez un taux de 20% de pertes supplémentaire :`ipfw pipe 1 config delay 20ms plr 0.2` Envoyez un paquet de 30000 octets à travers une nouvelle connexion TCP.

Comprendre et analyser précisément la courbe de l'évolution de séquence et la capture de paquets. Refaites des expériences si nécessaire.

2.7 Taille du buffer d'émission

Si nécessaire, relisez le cours sur la récupération d'erreur et des mécanismes à fenêtre d'anticipation (il n'est pas encore trop tard !).

- Supprimez les pertes sur M2.
- Après une ouverture de connexion TCP, réglez la taille du buffer d'émission à 1000 octets (options d'une socket dans socklab : commande `option`).

- Vérifiez que le délai ajouté sur la machine M2 est toujours de 20ms.
- Effectuez l'envoi de 10000 octets depuis la machine au buffer réduit.
- Capturez les paquets résultants.

- **Expliquez le flux de paquets que vous observez.**
- **Quelles est la différence avec un buffer de taille plus importante.**
- **Estimez par le calcul le débit (au niveau applicatif) que l'on obtient dans ce cas.**
- **Refaites l'expérience avec un buffer de 3000 octets. Conclusions ?**

2.8 Vérification expérimentale de l'influence du délai de transmission

- Supprimer sur M2 le délai supplémentaire. réglez la taille du buffer d'émission à 10000 octets (`sysctl net.inet.tcp.sendspace=10000`).
- Envoyez un flux continu TCP entre M1 et M3 (dans `/root/ipmt-tools` : `tcpmt -p Numeroport Host; tcptarget`.
- Observez le débit obtenu.
- Augmentez progressivement le délai sur M2 (1ms à 10ms) jusqu'à que le débit diminue. (on peut laisser tourner `tcpmt`).
- Vérifiez le RTT réel à l'aide de ping.

Expliquez pourquoi le débit diminue et par des calculs retrouvez à partir de quel délai il doit diminuer.

Pour s'aider : Capturez des paquets pendant 1 ou 2 secondes (attention cela va très vite !).

Comprendre et analyser précisément la courbe de l'évolution de séquence et la capture de paquets. Refaire si nécessaire des captures supplémentaires en faisant varier le délai sur M2

2.9 Vérification expérimentale de l'influence de la taille du buffer d'émission

Avec un temps de propagation de 10ms, estimez par le calcul la taille minimum du buffer d'émission pour que le débit applicatif soit maximal.

Vérifiez votre calcul expérimentalement. **ATTENTION** le buffer de réception (voir contrôle de flux) peut aussi limiter le débit si il est inférieur à celui d'émission.

- Fixez un délai sur M2 de 10ms.
  Modifiez sur M1 la taille du buffer d'émission de TCP avec le résultat de votre calcul (commande `sysctl`).
- Relancez `tcpmt` et observez le débit.
- Refaites l'expérience en faisant varier la taille du buffer d'émission au dessus et en dessous de la valeur initiale. Expliquez.

3 Contrôle de flux

3.1 principe

- Fixez sur M2 un délai de 20ms.
- Ouvrez une connexion TCP, entre M1 et M3.
- Fixez la taille du buffer de réception à 1000 octets. Attention la taille du buffer doit être fixée avant l'ouverture de la connexion.
- Emettez sur cette connexion deux messages de 2000 octets.
  Résumez sur un croquis temporel l'expérience en faisant apparaître les champs *sequence number*, *ack. number* et *window* de l'entête TCP.

Pourquoi l'émetteur continue-t-il à envoyer des paquets d'un octet de donnée ? Donnez l'évolution du temporisateur associé à ces émissions.

Faites ensuite des Read successifs coté récepteur de 500 octets.

Pourquoi le récepteur ne débloque-t-il pas tout de suite la situation ? A partir de quand le fait-il ? Quel intérêt ?

3.2 Influence de la taille du buffer de réception

Estimez par le calcul et par expérimentation la taille minimale du buffer de réception pour obtenir un débit maximal dans le cas d'un délai de transmission de 10 ms.

- **Rappelez à quoi servent les buffers d'émission et de réception dans le récupération d'erreur et le contrôle de flux.**
- **Est que l'utilisation d'un buffer d'émission de taille supérieure à celui de réception est pertinente ?**
- **Et le contraire ?**
- **Refaites des expériences si nécessaire.**

4 Algorithme de Nagle

Cette expérience permet de mettre en évidence l'algorithme développé par Nagle pour TCP dans le but de minimiser le nombre de paquet circulant sur le réseau. Il consiste à regrouper l'émission de paquets de petites tailles. TCP n'envoie pas plus d'un paquet de petite taille (plus petit qu'un MSS) sans avoir reçu d'acquittement. Pour observer ce phénomène, il faut donc faire varier le délai de transmission.

Lancez un `telnet -y`[1] entre M1 et M3. Pour deux délais différents sur M2 (0ms et 1000ms) :

- Effectuez une frappe très rapide de caractères (on peut utiliser la répétition automatique de la frappe)

[1] Le `-y` désactive le chiffrement du trafic.

- Analysez les paquets échangés.

Expliquez vos observations