UNIVERSITÉ Grenoble Alpes

Master 1 Informatique

Étude détaillée du protocole TCP
Le contrôle de congestion

P. Sicard, M. Heusse,

22 février 2023

Introduction

L'objectif de ce TP est de comprendre comment le protocole TCP (Transfert Control Protocol) gère le contrôle de congestion du réseau sur lequel il échange des paquets. Ce mécanisme s'ajoute à celui du contrôle de flux et de la récupération d'erreur vu dans un autre TP. Le contrôle de congestion consiste à essayer de gérer les problèmes d'embouteillages dans le réseau. Ces "embouteillages" se manifestent par un flux de paquet plus important en entrée qu'en sortie d'un carrefour (dans Internet un routeur). Les cas d'embouteillage se matérialisent par des pertes de paquets dans le réseau : destruction par les routeurs saturés et ne pouvant plus mémoriser les paquets entrant en attente de reémission.

Ces problèmes de congestion devrait dans l'idéal être résolu dans la couche réseau. Des solutions existent à ce niveau mais ne sont pas développées à grande échelle. C'est du coup dans TCP que des mécanismes permettant de réguler les trafics à l'émission ont été mis en place afin de palier aux congestions. Cela est d'autant plus délicat à ce niveau que TCP n'a aucune connaissance de la topologie du réseau et ne peut agir sur le fonctionnement des routeurs.
Il est à remarquer que le contrôle de congestion peut être vu comme un contrôle de flux où le récepteur serait le "réseau". De la même façon que pour le contrôle de flux, on va limiter l'émetteur grâce à une `fenêtre de congestion`. Suivant les capacités du réseau et du récepteur à un moment donné, ce sera l'un des deux contrôles (flux ou congestion) qui limitera le flux d'émission au niveau de TCP. La taille de la fenêtre de congestion est notée *cwnd*, celle du contrôle de flux *win*.

L'algorithmes de contrôle de congestion :

- **Slow start et TAHOE**
  Une première idée : "émettre de plus en plus jusqu'à l'observation d'une congestion, à partir de là essayer de réguler l'émission pour supprimer la congestion" Au niveau TCP , l'observation de la perte d'un paquet est assimilé à un début de congestion. Une perte de paquet est connu dans deux cas :

- Au moment du déclenchement du timer de re-émission.
- A la réception de quatre acquittements portant le même numéro. Permet d'anticiper la sonnerie du timer de reémission qui est souvent sur-dimensionné.

La figure 1 présente un exemple d'évolution de la fenêtre de congestion (en nombre de paquet). Le seuil de la fenêtre de congestion est fixé au départ à 64 koctets.

- Le mécanisme de *Slow Start* consiste à initialiser à 1 MTU puis à doubler la taille de la fenêtre à chaque émission (en fait la fénêtre est augmentée de 1 à chaque réception d'acquittement, ce qui revient à la doubler). On observe cette augmentation exponentielle dans la première partie de la courbe.
- En 1 un paquet est perdu, il y a une congestion. La taille de la fenêtre est remis à 1, le seuil est fixé à la taille de la fenêtre (au moment de la perte) divisée par deux.
- En 2 le seuil est atteint, la taille de la fenêtre augmente alors de façon linéaire.
- En 3 et 4 de nouvelle pertes se produisent.

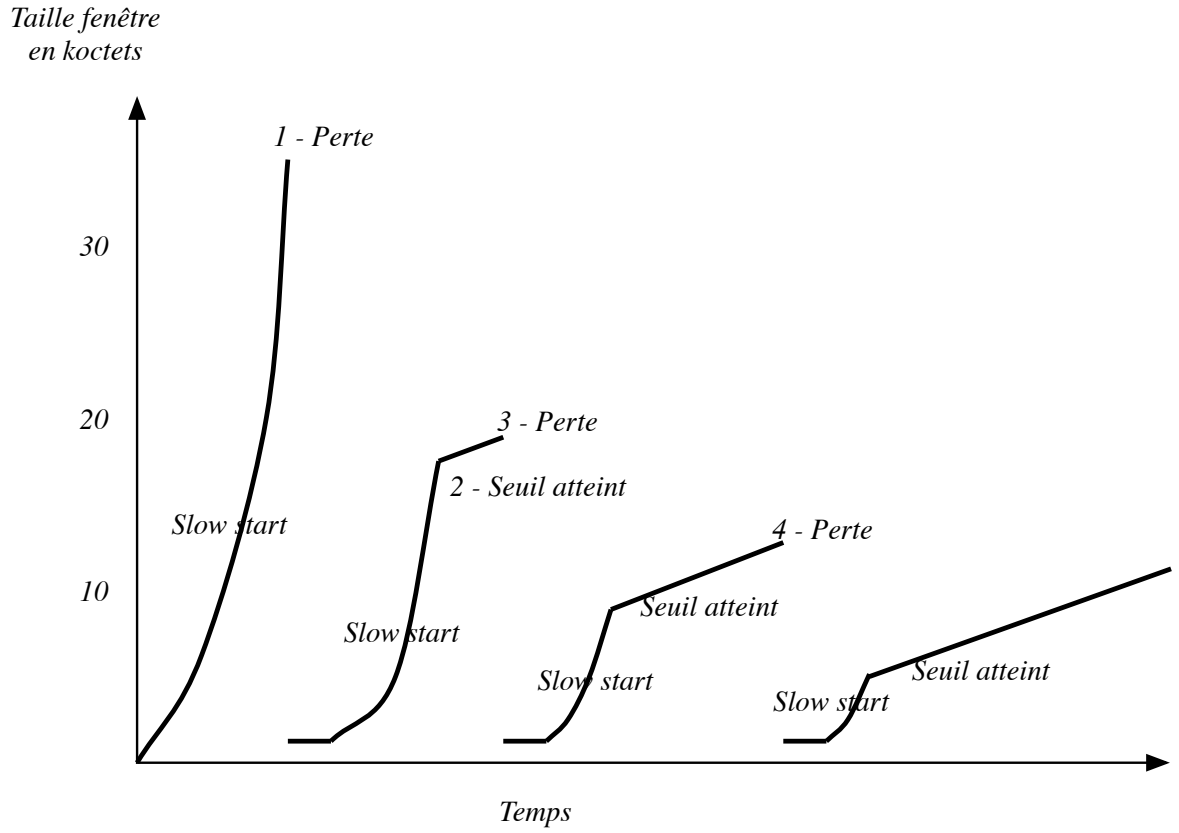

FIGURE 1 – Evolution de la fénêtre de congestion par le Slow start et TCP TAHOE

- **Amélioration par TCP *RENO* et *newRENO***

L'idée dans la version *RENO* du contrôle de congestion est de supposer que si des acquittements dupliqués arrivent après une perte, la congestion n'est pas importante (puisque les acquittements reviennent). Dans ce cas on ne repart pas en mode *slow start* après une perte de paquet reconnu à la réception d'acquittements dupliqués. Au moment de la réemission, le seuil prend la valeur de la moitié de la fênetre courante

et la fenêtre est divisée de moitié. Mais la fenêtre continue d'augmenter de 1 à la réception des acquittements dupliqués. Seul le paquet perdu est re-émis. Il n'y a pas d'attente importante puisque TCP peut continuer à émettre des paquets en attendant un acquittement non dupliqué. Cette phase est appelé "recouvrement rapide" (Fast recovery).
A la réception d'un acquittement non dupliqué la fenêtre reprend la valeur du seuil. La fenêtre continue alors à croître linéairement. La figure 2 montre l'évolution de la fenêtre avec ce mécanisme. Le seuil est noté *twnd* et la fenêtre *cwnd*. Dans le cas d'expiration de timer, le comportement est le même que Tahoe.

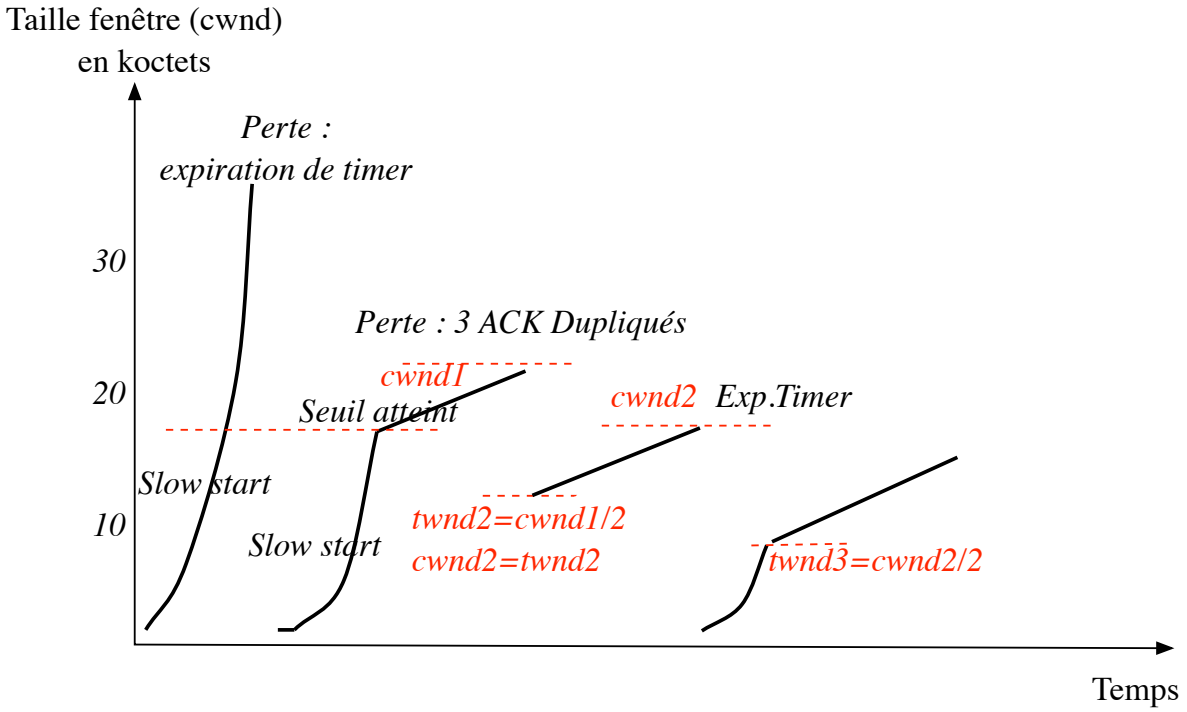

FIGURE 2 – Evolution de la fénêtre de congestion par TCP RENO

Une version appelée *new RENO* a été mise en place par la suite, elle affine le comportement de TCP dans le cas de plusieurs pertes successives (avec reconnaissance par acquittements dupliqués).

Déroulement du TP

- Montez le réseau de la figure 3. On utilisera des switchs 100 mégabits/s.
- Configurez les interfaces en jeu. Vérifiez que vous avez bien des réseaux en 100 mégabits/s. C'est le cas si vous avez utilisé des switchs.
  Sinon la commande `ifconfig em0 media 100baseTX mediaopt full-duplex` (à faire avant le branchement) permet de forcer le débit. Vérifiez que l'interface est alors bien en 100 mégabits/s et `full duplex`)
- 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.
- Lisez le mémento sur les utilitaires donné à la fin de ce document.

Dans l'ensemble des expériences suivantes on capturera les paquets sur M2 (sur l'interface connectée à M1).

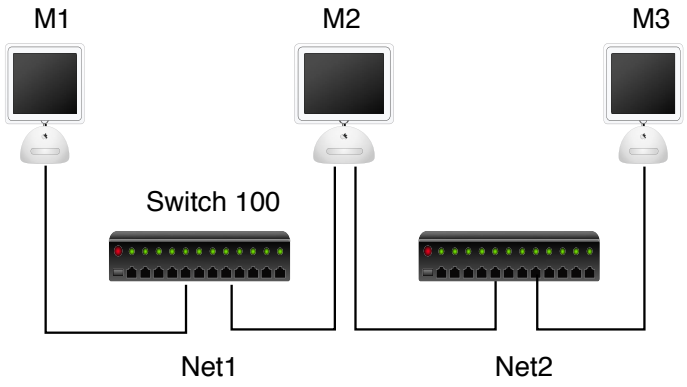

FIGURE 3 – Réseau à configurer

1 Contrôle de congestion

Sur les deux machines M1 et M3, réglez TCP comme suit :

- `sysctl net.inet.tcp` : liste des paramètre TCP
- `sysctl net.inet.tcp.hostcache.expire=0` : Avec cette option, TCP conserve en mémoire des paramètres pour une machine destinatrice (RTT, RTTvar, fenêtre de congestion...). Au passage on peut observer ces paramètres grâce à `sysctl net.inet.tcp.hostcache.list`
- `sysctl net.inet.tcp.sack.enable=0` , permet de désactiver le mécanisme d'acquittement sélectif possible dans TCP.
- `sysctl net.inet.tcp.delayed_ack=0` : cette option permet d'ajouter à TCP un délai avant émission d'un acquittement afin de minimiser le nombre d'acquittement. Nous le supprimons afin de faciliter l'analyse du contrôle de congestion à travers la capture des paquets.
- `sysctl net.inet.tcp.rfc3390=0`, désactive l'option de TCP qui ajuste la taille de la fenêtre initiale de congestion lors du slow start.
- `sysctl net.inet.tcp.sendbuf_auto=0`, désactive une option permettant à TCP d'ajuster la taille du buffer d'émission lors d'une connexion.
- `sysctl net.inet.tcp.recvbuf_auto=0`, désactive une option permettant à TCP d'ajuster la taille du buffer de réception lors d'une connexion.
- `sysctl net.inet.tcp.initcwnd_segments=1`, option permettant de définir la taille de la fenêtre initiale du slow start.

Sur la machine M2 réglez la traversée de IP à 20 ms (20 dans chaque sens pour obtenir un RTT de 40 ms).

Pour cela si nécessaire (absence de la commande `ipfw`) chargez tout d'abord le module *dummynet* (`kldload dummynet`)et lisez le passage sur *dummynet* en fin d'énoncé !

- `ipfw pipe 1 config delay 20`
- `ipfw add 4001 pipe 1 ip from any to any in`
- Ne pas oubliez la règle : `ipfw add 5000 permit ip from any to any`

1.1 Le *Slow Start*

Lancez Wireshark sur M1.
Emettez sur le machine M1 un flux continu à destination de M3 :
dans `/root/ipmt-tools` : `tcpmt -p Numeroport Host ; tcptarget -p Numeroport` .
Capturez les paquets (pendant quelques secondes) circulant à ce moment là et observez l'évolution des numéros de séquences sur la courbe donnée par `Wireshark` :

- 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.
- Il faut cliquer sur le bouton `Switch Direction`.
- 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 `WINDOW`.
- 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 dézoomer 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 clic droit de la souris (enfoncé plus déplacement).

- **Comprendre et analyser précisément cette courbe.**
- **Retrouvez le *Slow start* du contrôle de congestion de *TAHOE*.**
- **Une fois le *Slow Start* effectué, qu'est ce qui limite le flux de l'émetteur ?**
- **Retrouvez par le calcul le débit applicatif que vous observez. Il est à remarquer que la pente de la courbe est proportionnelle ce débit.**
- **A quelle taille la fenêtre de congestion est elle limitée ici. Pourquoi ?**

1.2 Fenêtre de congestion et de contrôle de flux

Sur le récepteur réglez le buffer de réception à 10000 octets. Recommencez la capture d'un flux de M1 vers M3.

Comprendre et analyser précisément la courbe obtenue.
Vérifiez sur la courbe que c'est la fenêtre du contrôle de flux qui limite le débit.
Retrouvez par le calcul le débit applicatif que vous observez. A quelle taille la fenêtre de congestion est elle limitée ici. Pourquoi ?

1.3 Cas de congestion

- Remettez les buffers d'émission et de réception à leur valeur d'origine. Réglez sur M2 un délai permettant d'obtenir un RTT de 20 ms.
- Retrouvez par le calcul le débit que l'on obtient entre M1 et M3 (à l'aide de `tcpmt`) avec ces buffers et ce RTT.
- Lancez un flux `tcpmt` entre M1 et M3, capturez les paquets, observez le débit obtenu (sans arrêter la capture).
- Sans trop attendre (nombre de paquets important capturés) réglez sur M2 un taux de pertes de paquet au moment de la traversée de IP : `ipfw pipe 1 config delay 5ms plr 0.02` (donne un taux de perte de 2 pour 100)
- Observez le débit moyen auquel on aboutit, arrêtez sans trop tarder la capture.

- Observez et analysez l'évolution du débit (sous *wireshark* : *graphique des flux TCP* puis *débit*).
- Analysez la courbe de l'évolution des numéros de séquence des paquets capturés, montrez et expliquez la différence de pente (proportionnelle au débit) sur les différentes parties de la courbe (sans ou avec les pertes).
- En regardant de plus près la courbe (zoom), comprendre et analyser précisément l'évolution de la taille de la fenêtre de congestion au moment des pertes de paquets, des re-émissions et des acquittements de ré-émission.
- Vérifiez et montrez sur la courbe les re-émissions dues à l'expiration d'un timer ou à la réception d'acquittements dupliqués. Dans chacun des cas, expliquez précisément l'évolution de la taille de la fenêtre de congestion.
- A votre avis est-ce la version *New RENO*, *RENO* ou *TAHOE* de TCP qui est utilisée ici. Argumentez. Refaites des expériences si nécessaire.
- Si c'est TCP Reno ou New Reno, montrez sur vos courbes les phases de ***fast recovery***.

- Comparez les débits entre M1 et M3 (à l'aide de `tcpmt`) en faisant varier le taux de pertes sur M2 (0.01, 0.02, 0.05 et 0.1).
- Tirez des conclusions sur ce type de contrôle de congestion.

2 Un vrai cas de congestion

- Rajouter un quatrième ordinateur M4 sur le réseau de M1. Configurez son interface réseau et sa table de routage.
- Sur cet ordinateur mettre à jour les options TCP suivantes : `sysctl net.inet.tcp.hostcache.exp`
  `sysctl net.inet.tcp.delayed_ack=0`
  `sysctl net.inet.tcp.rfc3390=0`

- `sysctl net.inet.tcp.sendbuf_auto=0`
  `sysctl net.inet.tcp.initcwnd_segments=1`

- Vérifiez le débit obtenu entre M1 et M3 et entre M4 et M3 à l'aide de *tcpmt*.
  Si le débit est inférieur à 50 mégabits/s, diminuez le délai sur M2 ou augmentez la

taille du buffer d'émission sur M1 et M4.
- Lancer *wireshark* sur M1 et M4.
- Lancer en même temps deux flux TCP à l'aide de *tcpmt* sur M1 et M4 à destination de M3 (avec deux *tcptarget* sur des ports différents).
- Arrêtez les captures de paquets sur M1 et M4 dès que les débits sont à peu près stabilisés.

- **Observez l'évolution des débits sur les deux machines.**
- **Ces débits sont-ils stables ? Pourquoi ?**
- **Ces débits sont-ils équitables ? Pourquoi ?**
- **Analysez les captures à l'aide des courbes de numéros de séquence. Montrez sur la courbe issue de wireshark les pertes de paquets et les diminutions des débits qui en résultent.**
- **Où a lieu la congestion ?**

3 Algorithme de Nagle

Si il vous reste du temps...

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 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. On peut utiliser le compte *guest* pour cela. 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)
- Analysez les paquets échangés.

Expliquez vos observations en rappelant l'algorithme de Nagle. Quel intérêt ? Cela peut il poser problème pour certaines applications ?

Rappel sur les 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 primitives de base de manipulation des sockets.

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

- `tcpmt, tcptarget` : Logiciels client/serveur vous permettant de générer des flux continus au dessus de TCP (dans le répertoire `/root/ipmt-tools`).
- `wireshark` : 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` Options du noyau système permettant de modifier le comportement d'une machine au niveau du protocole IP (délai, pertes, ...). 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. Le `in` spécifie en entrée de la couche IP, on peut spécifier `out`. Si on ne spécifie rien, le délai est rajouté deux fois (en entrée et en sortie).
  6. Suppression d'une règle : `ipfw delete 4001`
  7. Attention les règles sont "testées" suivant l'ordre croissant de leur numéro.
  8. ATTENTION la granularité du délai n'est pas forcément de la finesse désirée, testez par un ping pour vérifiez le RTT obtenu.
- `sysctl` Gestionnaire de paramètres du système et en particulier paramètres de TCP (taille des buffers, durée de timer ...)
  - `sysctl net.inet.tcp` (liste des paramètre TCP)
  - `sysctl net.inet.tcp.sendspace=32768` (modification du buffer d'émission par exemple)