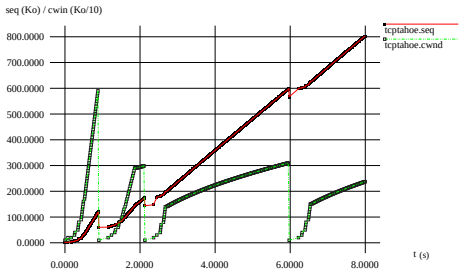
ARES - Lab n°5

Couche transport (2) : Contrôle de congestion TCP



1 Contrôle de congestion (sans machines)

TCP est utilisé pour le transport fiable de données dans l'Internet. Nous avons précédemment étudié la gestion des connexions et les mécanismes TCP. Dans les exercices suivants, nous allons nous intéresser à un autre comportement fondamental de TCP : le contrôle de congestion.

1.1 Détection de la congestion

La conception de TCP date de la fin des années 70. Plusieurs algorithmes de contrôle de congestion ont été intégrés depuis, principalement suite aux travaux de Van Jacobson publiés en 1988. Ces derniers continuent à évoluer dans les différentes variantes de TCP. Les exercices proposés dans la suite sont fondés sur les dernières mises à jour : le RFC 5681 de septembre 2009.

1. Pour TCP, quel phénomène indique une congestion dans le réseau ?
2. Que se passe-t-il dans un routeur pour susciter ce phénomène ?
3. Pour TCP, ce phénomène permet de déduire la congestion. Mais celui-ci peut aussi se produire quand il n'y a pas de congestion dans le réseau. Dans quels autres cas un tel phénomène peut-il apparaître ?
4. Si ce phénomène n'indique pas toujours une congestion, pourquoi TCP se base-t-il sur cette inférence ? Pourquoi n'utilise-t-on pas une approche où le routeur constatant la congestion envoie un message explicite à l'émetteur ?

1.2 Algorithmes de contrôle de congestion

Pour le contrôle de congestion, TCP utilise un seuil qui indique le débit au-dessus duquel la congestion risque de se produire. Ce seuil est exprimé par le paramètre `ssthresh` (en octets). Pour obtenir le débit seuil on divise `ssthresh` par le RTT (*Round Trip Time*). Le débit peut varier en-dessous et au-dessus du seuil `ssthresh`$/RTT$. L'émetteur maintient un second paramètre, `cwnd` (taille de la fenêtre de congestion) qui indique le nombre maximum d'octets qu'il peut envoyer avant de recevoir un acquittement. Quand `cwnd > ssthresh`, l'émetteur fait particulièrement attention à ne pas provoquer de congestion.

1. Supposons que `ssthresh` soit à 5000 octets, `cwnd` est à 6000 octets, et la taille d'un segment est de 500 octets. Un émetteur envoie douze segments de 500 octets dans une période `RTT`, et reçoit douze acquittements (un pour chaque segment). Que deviennent les valeurs de `ssthresh` et `cwnd` ? Comment s'appellent ces changements de valeurs ?
2. Supposons que `ssthresh` soit toujours à 5000 octets, que `cwnd` est maintenant à 14.000 octets, que l'émetteur envoie $14.000/500 = 28$ segments, et que l'émetteur reçoive une indication de congestion avant de recevoir le premier acquittement. Que deviennent les valeurs de `ssthresh` et `cwnd` ? Comment s'appellent ces changements de valeurs ?
3. Nous venons de voir comment augmente et diminue `cwnd` en fonction de l'absence ou la présence d'indicateurs de congestion. Comment s'appelle cet algorithme ? Sur quel principe repose cet algorithme ?
4. Au démarrage, ou après avoir reçu une indication de congestion, la valeur de `cwnd` est plus petite que la valeur de `ssthresh`. Décrivez la manière permettant d'augmenter `cwnd` quand celle-ci est inférieure à `ssthresh`, en fonction de l'exemple suivant. Supposons que `ssthresh` soit égal à 3000 octets et que `cwnd` soit égal à 500 octets, la taille d'un segment. L'émetteur a plusieurs segments prêts à être envoyés. Combien de segments envoie l'émetteur pendant la première

période RTT ? S'il reçoit des acquittements pour tous ses segments, que devient la valeur de `cwnd` ? Combien de segments envoie l'émetteur pendant la deuxième période RTT ? S'il reçoit des acquittements pour tous ses segments, que devient la valeur de `cwnd` ? En général, comment évolue la taille de `cwnd` ?

5. Comment s'appelle la période pendant laquelle `cwnd` est plus petit que `ssthresh` ?
6. Que devient la valeur de `ssthresh` si l'émetteur reçoit une indication de congestion pendant que `cwnd` est plus petit que `ssthresh` ?

1.3 Débit moyen d'une connexion TCP

Supposons que nous souhaitions effectuer un transfert de données de taille importante à travers une connexion TCP.

1. En négligeant la période pendant laquelle `cwnd` est plus petit que `ssthresh`, montrez que le débit moyen d associé à une connexion TCP est égal à :

$$d = \frac{3}{4}\frac{W * MSS}{RTT}$$

où W est la taille de la fenêtre (en segment) au moment de la congestion, MSS la taille d'un segment (supposée maximale), et RTT est le délai aller-retour (supposé constant durant la période de la transmission).

2. Montrez que le taux de pertes p est égal à :

$$p = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

3. Montrez que si le taux de pertes observé par une connexion TCP est p, alors son débit moyen d peut être approximé par :

$$d = \frac{1,22 * MSS}{RTT\sqrt{p}}$$

4. Quels autres paramètres peuvent influer sur le débit d'une connexion TCP ?
5. Quelle utilité voyez-vous à la relation calculée dans la dernière formule de d ?

2 Etude de la latence d'un serveur web (sans machine)

Nous souhaitons étudier la latence liée à la réponse à une requête HTTP[1]. Nous faisons les hypothèses simplificatrices suivantes :

- Le réseau n'est pas congestionné (pas de pertes ni de retransmissions) ;
- Le récepteur est doté de tampons de réception infinis (limitation de l'émetteur uniquement liée à la fenêtre de congestion) ;
- La taille de l'objet à recevoir du serveur est O, un multiple entier du MSS (MSS à pour taille S bits) ;
- Le débit de la liaison connectant le client au serveur est R (bits/s) et on néglige la taille de tous les entêtes (TCP, IP et liaison). Seuls les segments transportant des données ont un temps de transmission significatif. Le temps de transmission des segments de contrôle (ACK, SYN...) est négligeable ;
- La valeur du seuil initial du contrôle de congestion n'est jamais atteinte ;
- La valeur du délai aller-retour est RTT.

1. Dans un premier temps, nous supposons que nous n'avons pas de fenêtre de contrôle de congestion. Dans ce cas montrez que la latence L peut s'exprimer de la manière suivante :

$$L = 2RTT + O/R$$

[1] Latence d'une requête HTTP : laps de temps pour la création de la connexion et la récupération de l'intégralité de l'objet demandé.

2. Nous supposons maintenant une fenêtre de congestion **statique** de taille fixe égale à W. Calculez la latence dans ce premier cas :

$$WS/R < RTT + S/R$$

3. Nous supposons toujours une fenêtre de congestion **statique** de taille fixe égale à W. Calculez la latence dans ce second cas :

$$WS/R > RTT + S/R$$

4. Comparez la latence obtenue avec une fenêtre de contrôle de congestion **dynamique** (*slow-start*) avec celle sans contrôle de congestion.

5. Application numérique :

R	O/R	L (sans *slow start*)	K'	L (latence globale de TCP)
56 Kbps				
512 Kbps				
8 Mbps				
100 Mbps				

K' est le nombre de fenêtres envoyées avant de passer au second cas ($log2(1 + RTT * R/S)$). Considérez trois cas :

(a) S= 512 octets, RTT= 100 msec, O=**100 Koctets** (=200S) ;

(b) S= 512 octets, RTT= 100 msec, O=**5 Koctets** (=10S) ;

(c) S= 512 octets, RTT= **1 seconde**, O=5 Koctets (=10S).

3 Analyse des mécanismes TCP

Voici quatre captures de trafic HTTP. Les trois premières sont pré-enregistrées et reposent sur des clients et serveurs éloignés (les trois sont réalisées avec une sonde proche du client). Vous réaliserez la dernière capture dans un environnement local (sur la plate-forme d'expérimentation). Pour chacune d'elles, tracez précisément le chronogramme et étudiez les mécanismes de contrôle de congestion mis-en-œuvre. Discutez en particulier des points suivants :

1. Quel est le RTT moyen ?
2. Reconnaissez-vous les mécanismes de contrôle de congestion ?
3. Jusqu'à combien de segments sont transmis par RTT ?
4. Quel est le débit moyen alors atteint ?
5. Un envoi continu apparaît-il ?
6. Des perturbations sont-elles présentes (déséquencement, retransmission...) ?

3.1 Trafic HTTP Paris-Brisbane (WAN Intercontinental : 17000km)

Utilisez le logiciel `wireshark` (sans avoir besoin des droits d'administrateur) sur le poste ARI pour cette première trace. Télécharger la trace `tme5-wau.dmp` (similaire à celle capturée précédemment) soit à partir du répertoire `/Infos/lmd/2012/master/ue/ares-2012oct`, soit sur la page web `http://www-rp.lip6.fr/~fourmaux/Traces/labV6.html`.

3.2 Trafic HTTP Paris-Budapest (WAN Continental : 1200km)

Utilisez le logiciel `wireshark` (sans avoir besoin des droits d'administrateur) sur le poste ARI pour cette trace. Télécharger la trace `tme5-whu.dmp` (similaire à celle capturée précédemment) des emplacement utilisés précédemnets

3.3 Trafic HTTP Paris-Evry (MAN : 36km)

Utilisez le logiciel `wireshark` (sans avoir besoin des droits d'administrateur) sur le poste ARI pour cette trace. Télécharger la dernière trace `tme5-man.dmp` (similaire à celle capturée précédemment) des emplacement utilisés précédemnets

3.4 Trafic HTTP local (LAN)

3.4.1 Capture du trafic TCP résultant d'un échange HTTP local

Cette première capture a pour but de percevoir la nature du trafic TCP dans un LAN. Sur la plateforme d'expérimentation, la topologie 1 a été configurée (postes clients et serveur sur le même LAN). Réalisez la capture de trafic TELNET à l'aide du logiciel `wireshark` :

- A partir du poste ARI **N**, connectez-vous sur les 3 VM correspondantes de la plateforme à l'aide de 3 fenêtres textuelles
  - fenêtre 1 (hôte "client") : tapez `ssh -X etudiant@10.0.7.`**N**1
  - fenêtre 2 (hôte "sonde") : tapez `ssh -X root@10.0.7.`**N**2 (attention, vous êtes administrateur)
  - fenêtre 3 (hôte "serveur") : tapez `ssh -X etudiant@10.0.7.`**N**3
- Vérifiez qu'un gros fichier est accessible depuis le répertoire `public_html` du compte `etudiant` (à créer avec les droit d'accès publics si absent) et que le serveur HTTP tourne sur la VM `10.0.7.`**N**3 (fenêtre 3)
  - générez un gros fichier, tapez : `dd if=/dev/zero of=public_html/fichier100Mo bs=1M count=100`
  - vérifiez l'accès public du fichier(`-rw-r--r--`), taper : `ls -l public_html/fichier100Mo`
  - vérifiez que le serveur HTTP tourne (`apache2`)
  - visualisez la configuration des interfaces, en particulier celle sur le LAN d'étude pour vérifier l'adresse IP du serveur pour la connexion du client (devrait être `10.`**N**`.1.`**N**3)
- Démarrez la capture en lançant l'analyseur sur `10.0.7.`**N**2 (fenêtre 2)
  - lancez l'analyseur, tappez : `wireshark`
  - initier la capture sur l'interface `eth1`, comme indiqué précédemment
- Démarrez un client HTTP sur `10.0.7.`**N**1 (fenêtre 1)
  - démarrez le client HTTP de votre choix (`firefox`...)
  - tapez l'URL suivante `http://10.`**N**`.1.`**N**`3/~etudiant/fichier100Mo`
- Observez la capture se réaliser dans la fenêtre de `wireshark`
- Terminez la capture. **Filtrez le trafic afin de ne conserver que celui relatif à TCP** (filtre = `tcp`). Enregistrez la trace filtrée pour pouvoir la ré-utiliser ultérieurement. Ne quittez pas l'application et répondez aux mêmes questions que pour les trois traces pré-enregistrées précédentes.

3.4.2 Sans la plateforme...

En cas de problème d'accès à la plateforme d'expérimentation ou si vous souhaitez réviser sur une autre machine, vous pouvez télécharger la trace `tme5-lan.dmp` (similaire à celle capturée précédemment) soit à partir du répertoire `/Infos/lmd/2012/master/ue/ares-2012oct`, soit sur la page web `http://www-rp.lip6.fr/~fourmaux/Traces/labV6.html`, puis l'analyser avec le logiciel `wireshark` (sans avoir besoin des droits d'administrateur).

4 Avant de quitter la salle

- Détruisez le gros fichier (`rm public_html/fichier100Mo`).
- Si vous avez enregistré des captures sur la VM "sonde", n'oubliez pas de les rapatrier sur votre compte utilisateur de l'ARI. Tapez : `scp root@10.0.7.`**N**`2:<ma_trace> <destination_locale>`.
- Avant de terminer vos connexions sur les équipements de la plateforme, supprimez vos fichiers et modifications effectuées afin de retrouver l'état initial.