M1 RES - Travaux dirigés 4/10

Transport : Contrôle de congestion TCP

1 Protocole TCP (suite)

TCP est utilisé pour le transport fiable de données dans l'Internet. Nous avons précédemment étudié la gestion des connexions et mécanismes TCP. Dans les exercices suivants, nour 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 ajoutés 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 : les RFC 2581 et 2582 d'avril 1999 (TCP newReno).

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 est-ce que TCP se base sur cette inférence ? Pourquoi n'utilise-t-on pas une approche où le routeur constatant la congestion envoi 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 paramètre `cwnd` (*Congestion WiNDow*) qui indique le nombre 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 paquet est de 500 octets. Un émetteur envoi douze paquets de 500 octets dans une période `RTT`, et reçoit douze acquittements (un pour chaque paquet). 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 envoi $14.000/500 = 28$ paquets, 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, et 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 paquet. L'émetteur a plusieurs paquets prêts à être envoyés. Combien de paquets envoie l'émetteur pendant la première période *RTT* ? S'il reçoit des acquittements pour tous ses paquets, que devient la valeur de `cwnd` ? Combien de paquets envoie l'émetteur pendant la deuxième période *RTT* ? S'il reçoit des acquittements pour tous ses paquets, 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. Montrer que le taux de pertes p est égal à :
$$p = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$
3. Montrer 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 (*faculatif*)

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 (bps) 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 $L = 2RTT + O/R$.
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	Latence sans S.S.	K'	Latence 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).

[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.