Correction Contrôle X99 Réseaux et protocoles
INF 586

# Question 1.

- Un lien en A et B à 1 Mbps. En 1 seconde, on transmet $10^6$ bits, la transmission de $8.10^7$ bits prendra Tt = 80 secondes. Débit effectif = 1 Mbps. Pas d'ACK car commutation de circuit, donc à priori pas d'erreur.

- 5. Pas de transmission continue car la fenêtre = 1, et le délai de propagation + délai de transmission n'est pas nul.

Le temps total de transmission du fichier Tt = Nb (nombre de paquets) x Tp (temps d'envoi d'un paquet et de réception d'un ACK.)
Nb = $8.10^7$ / P.
Tp = Dp + $Dtrx_{DT}$ + Dp + $Dtrx_{ACK}$.
Dp sec = $5.10^{-6}$.D (Km)
$Dtrx_{DT}$ = (P+80)/C
$Dtrx_{ACK}$ = 88/C

Application numérique:
1. D = 1 Km, C = $10^6$ bps, P = 256 bits. Nb = 312500, Dp = $5.10^{-6}$ sec
   Tp = $5.10^{-6}$ + (256+80).$10^{-6}$ + $5.10^{-6}$ + $88.10^{-6}$ = $434.10^{-6}$ sec.
   Tt = 135.625 secondes. Débit effectif = $8.10^7$ / 135.625 = **0.58986175 Mbps** ( < 1 Mbps)
2. D = 1 Km, C = $10^7$ bps, P = 256 bits. Nb = 312500, Dp = $5.10^{-6}$ sec
   Tp = $5.10^{-6}$ + $336.10^{-7}$ + $5.10^{-6}$ + $88.10^{-7}$ = 52.4 $10^{-6}$ sec
   Tt = 16.375 secondes, Débit effectif = **4.8854962 Mbps** (< 10 Mbps)
3. D = 10 Km, C = $10^6$ bps, P = 256 bits. Nb = 312500, Dp = $5.10^{-5}$ sec
   Tp = $5.10^{-5}$ + $336.10^{-6}$ + $5.10^{-5}$ + $88.10^{-6}$ = $524.10^{-6}$ sec
   Tt = 163.75 secondes, Débit effectif = **0.48854962 Mbps** (< 1 Mbps)
4. D = 1 Km, C = 50.106 bps, P = 10000 bits. Nb = 8000, Dp = $5.10^{-6}$
   Tp = $5.10^{-6}$ + 10080/50.$10^{-6}$ + $5.10^{-6}$ +88/50.$10^{-6}$ = 213.36 .$10^{-6}$
   Tt = 213.36 $10^{-6}$ x 8000 = 1.70688 secondes. Débit effectif = 8.107/1.70688 = **46.869141 Mbps** (<50 Mbps), mais l'efficacité est plus grande.

Erreur dans le raisonnement:
1. la surcharge provient en fait du temps d'établissement et de libération du circuit, ainsi que de certains bit de "framing".
2. Le trafic de données est sporadique, et même si on perd à cause des en-têtes, on gagne d'autre part à cause du multiplexage statistique.

# Question 2.

a) L'émetteur envoie des paquets de données numérotés. Le récepteur détecte les pertes en vérifiant les numéros de séquence. Si l'émetteur n'a pas de données "fraîches" à envoyer, il émet un "heartbeat" afin de détecter la perte éventuelle du (des) dernier (s) paquet (s) de données. EN cas de détection de paquet perdu, le destinataire demande par un NACK la retransmission du paquet à la source (en fait il le demande à un "serveur" chargé de la retransmission par la source).

L'adaptation du heartbeat permet de réduire la charge du réseau provenant des paquets heartbeat. Après un changement de terrain (données fraîches) on utilise un MaxIT faible, puis on l'augmente progressivement (de façon exponentielle). On s'arrête à une valeur maximale. Ceci permet une détection rapide des pertes de paquets éventuelles, tout en maintenant un charge relativement faibles des paquets heartbeat. L'intérêt d'adapter le MaxIT est donc de diminuer la charge en cas d'activité réduite (ce qui correspond à la nature de l'application, car les changements de terrain n'ont pas lieu très fréquemment).

b) Le serveur de fiabilité décharge la source de cette tâche, mais pose le problème de la localisation du serveur, ainsi que celui d'assurer la fiabilité entre la source et le serveur (ACK explicite).
La mise en place de serveurs secondaires permet de diminuer la charge du serveur primaire, par l'agrégation des demandes de retransmissions en provenance des serveurs secondaires, et aussi d'effectuer une retransmission locale au cas où le serveur secondaire dispose du paquet dont la retransmission est demandée. Inconvénients: architecture arborescente à calquer sur le réseau.

c) SRM vs. LBRM.
LBRM impose une hiérarchie, donc (+) permet de réduire les délais de retransmissions, mais (-) a un coût d'établissement.
SRM n'impose pas de structure hiérarchique, les demandes de retransmissions et les retransmissions elles-mêmes se font en multicast. C'est plus "fiable" dans le sens tolérance aux pannes…

Pour une explication détaillée sur ce thème voir http://citeseer.nj.nec.com/holbrook95logbased.html

# Question 3.

a) Si l'application est adaptative, on aura une diminution "contrôlée" de la qualité (1 image sur deux par exemple au lieu d'avoir une trame perdue de chaque image de deux trames). Pour nuancer ces propos: si FIFO on peut subir l'impact des autres et il faudrait que toutes les sources s'adaptent (TCP-friendliness). Si FQ: on peut s'adapter seul indépendamment du comportement des autres.

b)
B1): Flux UDP NON adaptatif, FIFO:

- Si débitUDP < 128 Kbps, pas de perte pour UDP, TCP s'adapte et consomme 1.5 Mbps - 128 Kbps.
- Si 128 K < débitUDP < 1.5 M, pertes UDP à R2, TCP obtient 1.5M - débitUDP.
- Si 1.5 M < débitUDP, pertes UDP à R1, débit TCP = 0;
- Si 10 M < débitUDP, pertes UDP à la source S2, R1 et R2. TCP fermé.

B2): Flux UDP NON adaptatif, FQ:

- Si débitUDP < 128 Kbps, tout est OK.
- Si 128 K < débitUDP < .75 M, pertes UDP à R2, TCP obtient 1.5M - débitUDP.
- Si .75 M < débitUDP < 1.5 M, pertes UDP à R2, TCP obtient 750 Kbps.
- Si 1.5 M < débitUDP, pertes UDP à R1, débit TCP = 750 Kbps;
- Si 10 M < débitUDP, pertes UDP à la source S2, R1 et R2. TCP = 750 Kbps.

B3) Flux UDP adaptatif, FIFO:
Si l'agressivité du flux est autant que TCP => partage équitable.

Sinon, UDP obtient plus.
B4) Flux UDP adaptatif, FQ:
L'équité est imposée par le réseau. L'adaptation apporte un gain individuel à UDP (voir a) ci-haut).

c) Si m flux UDP => dans le cas FIFO c'est le débit total qui compte.
Dans le cas FQ, chaque flux aura une "part" équitable des ressources. => pas d'équité envers un flux TCP (si un utilisateur malicieux ouvre plusieurs connexions UDP, il peut contourner le mécanisme FQ).

d) on peut jeter des paquets de façon incrémentale en fonction du dépassement par rapport au "fair share".

# Question 4.

Tous les arbres basés sur la source sont confondus avec l'arbre partagé dans cet exemple, car il n'y a pas beaucoup de "liens" dans le réseau. La modification du Mbone vers plus de connectivité aboutit à la possibilité d'avoir des arbres basés sur la source différents des arbres partagés.

DVMRP et PIM-SM: question de cours sur le multicast.

# Question 5.

Impact:
1. délai élevé -> établissement de connexion lent et aussi la récupération après une perte est lente.
2. Produit délai x bande passante est élevé -> inefficacité des mécanismes de retransmissions go-back-N.
3. Taux d'erreur élevé -> dégrade le débit
4. Bande passante asymétrique -> les ACKs peuvent ralentir le transfert!

Slow start:
Si Winit > 1, le débit pourrait être amélioré, mais on risque d'avoir une congestion dans le réseau, car le lien satellite n'est pas le seul lien entre la source et la destination.

MTU:
Si détection de MTU -> diminue le nombre de paquets à transmettre, ce qui améliore les performances en cas de perte de paquet.

Débit maximum = $2^{16}$ x MTU / 0.5

On peut pallier cette limitation par une extension de la fenêtre.

Si plusieurs pertes pendant une "fenêtre" le fast retransmit ne marche plus, et on aura un retard important à cause du fait qu'il faudrait attendre un RTT par paquet retransmis.
On pourrait utiliser une option de Selective Acknowledgment explicite. Pour compenser le fait que le fast retransmit ne permet plus d'inférer quel paquet est perdu.

Asymétrie:

Les acks peuvent être perdus. On peut effectuer un Ack Filtering puis ACK reconstruction. Même si la reconstruction n'est pas primordiale car les ACKs sont cumulatifs, la façon avec laquelle les ACKs arrivent à la source a un impact important sur le débit de la source car cela déclenche les nouveaux paquets à émettre. Il faudrait alors veiller au fait que la reconstruction d'ACK ne perturbe pas le self-clocking de TCP.