Université de Limoges | FACULTÉ DES SCIENCES ET TECHNIQUES

Master 1ère année

Réseaux Avancés I

Partiel — novembre 2013

Durée : 2h — Documents autorisés

Protocole temps réel, RTT et fragmentation – 7 points

1 – 4pts Une conférence va être retransmise sur Internet à l'aide d'un protocole temps réel basé sur UDP. Les paramétres sont les suivants :

- ⋆ L'algorithme de numérisation utilisé pour la vidéo et le son permet de produire un paquet de données pour une durée d'utilisation de $30ms$.
- ⋆ On a relevé les mesures suivantes avec la commande « `ping` » juste avant le début de la conférence :

```
pef@maconf:~$ ping recepteur.qqparten.fr
PING recepteur.qqparten.fr (1.2.3.4): 56 data bytes
64 bytes from 1.2.3.4: icmp_seq=0 ttl=57 time=84 ms
64 bytes from 1.2.3.4: icmp_seq=1 ttl=57 time=90 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=57 time=80 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=57 time=70 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=57 time=76 ms
```

- ⋆ On considérera que la conférence commence au temps t_0.
- ⋆ On veut que la retransmission débute le plus vite possible.

Questions :

a. Qu'est-ce que mesure la commande « `ping` » et comment déduire la latence, « latency », et la gigue, « jitter » ?
L'utilitaire « `ping` » mesure le RTT, c-à-d $2 * latence$*, la gigue mesure la variation de latence :* $gigue = ancienne_{latence} - nouvelle_{latence}$

b. Calculez la latence et le jitter sur les valeurs fournies par le « `ping` ».

RTT	*latence*	*gigue*
84	*42*	*0*
90	*45*	*3*
80	*40*	*5*
70	*35*	*5*
76	*38*	*3*

c. Par rapport à t_0 et aux caractéristiques de l'algorithme de numérisation employé, donnez :
   - ◇ le rythme d'envoi des paquets UDP produit pour la retransmission de la conférence ;
     Le rythme d'envoi correspond au rythme d'échantillonnage, c-à-d toutes les $30ms$.
   - ◇ le temps d'attente avant que les données du paquet UDP soient utilisées sur le récepteur, ainsi que le rythme d'utilisation de ces données ;
     Le temps d'attente doit être supérieur à la latence moyenne plus le max de la gigue : ici, $(42 + 45 + 40 + 35 + 38)/5 = 200/5 = 40$*, soit un temps d'attente de* $45ms$ *entre le début de la conférence et sa diffusion sur le récepteur* $t_0 + 45ms$.

d. Expliquez pourquoi certaines données ne pourront pas être utilisées lors de la réception.
Les données reçues sont utilisées au rythme d'un contenu toutes les $30ms$*. Si un paquet est reçu après son temps d'utilisation prévu, alors il sera ignoré et son contenu considéré comme perdu.*

2 – Un datagramme UDP est envoyé :
3pts
- ⋆ d'un réseau A dont la MTU est de 1500 octets pour le contenu de la trame (c-à-d que le datagramme IP le plus grand qui peut y circuler a une taille de 1500 octets) ;
- ⋆ dans un réseau B dont la MTU est de 1000 octets pour le contenu de la trame (c-à-d que le datagramme IP le plus grand qui peut y circuler a une taille de 1000 octets) ;

Questions :

a. Quel est la taille maximale des données que peut contenir un paquet UDP dans le réseau A ?
1500-20-8 = 1472

b. Comment va être fragmenté un paquet UDP de taille maximale lors de son arrivée dans le réseau B ? *Taille maximale des données dans le datagramme IP dans B : 1000 - 20 = 980*
Soient 1480 octets à fragmenter en 976 (divisible par 8) et 504 (divisible par 8).
Soient les offset de 0 pour le premier et de $976/8 = 122$ *pour le deuxième.*

Code détecteur et correcteur d'erreurs — 3 points

3 – Les transmissions d'une veille sonde spatiale utilisant la méthode de Hamming ont été récupérées lors
3pts de la rénovation d'un bâtiment de l'Agence spatiale européenne.
Le logiciel de traitement des données a été perdu, et seul subsiste une impression du résultat obtenu par ce logiciel sur deux séquences reçues :

Séquence reçue 1	11010110011100110	OK
Séquence reçue 2	00001010101010111	???

Questions :

a. Pouvez vous aider les ingénieurs chargés de réécrire le logiciel à déterminer si la méthode de Hamming a été utilisée en numérotant les bits de la séquence de « gauche à droite » ou de « droite à gauche » ?
On va calculer la méthode de Hamming sur les données qui sont indiquées comme justes, la « Séquence reçue 1 », soit en prenant la séquence dans le sens de lecture soit suivant le sens inversé :

11010110011100110	*donne une erreur sur le bit 20 ⟶ Ce n'est pas la méthode de Hamming !*
01100111001101011	*ne donne pas d'erreur : il y a un inversion des bits par rapport à la méthode vue en cours.*

La méthode a été utilisée suivant une lecture de « droite à gauche ».

b. Pouvez vous fournir les données reçues à partir de ces deux séquences ?
On va utiliser la même méthode sur la « Séquence reçue 2 » :

On inverse la séquence 2	*11101010101010000*
On calcule la méthode de Hamming	*erreur sur le bit 13*
On corrige le bit 13	*11101010101000000*

Les données peuvent être retrouvées en supprimant les 5 bits de contrôle des deux séquences ($m + k + 1 \leq 2^k$ avec $18 \leq 2^5$) :

Séquence 1	*101100110101*
Séquence 2	*110110100000*

Analyse de trame — 5 points

4 – Au même endroit du réseau, on a capturé les deux trames suivantes :
5pts

```
0000   00 26 BB 15 8E C5 C0 8C  60 00 BA 30 08 06 00 01   .&......‘..0....
0010   08 00 06 04 00 02 C0 8C  60 00 BA 30 C1 32 0A FE   ........‘..0.2..
0020   00 26 BB 15 8E C5 C1 32  0B 20                     .&.....2.
```

Ce qui donne :
```
<Ether  dst=00:26:bb:15:8e:c5 src=c0:8c:60:00:ba:30 type=0x806
|<ARP  hwtype=0x1 ptype=0x800 hwlen=6 plen=4 op=is-at hwsrc=c0:8c:60:00:ba:30
psrc=193.50.10.254 hwdst=00:26:bb:15:8e:c5 pdst=193.50.11.32 |
```

```
0000   00 26 BB 15 8E C5 C0 8C  60 00 BA 30 08 00 45 00   .&......‘..0..E.
0010   00 28 00 2B 00 00 40 06  FC EA 05 C7 AB A1 C1 32   .(.+..@........2
0020   0B 20 00 50 B2 41 00 0B  FB AB 00 00 00 00 50 12   . .P.A........P.
0030   20 00 63 CF 00 00                                   .c...
```

Que pouvez vous apprendre d'après le contenu de ces trames sur la configuration du réseau, les matériels qui communiquent entre eux et les services utilisés ?
Ce qui donne : `<Ether  dst=00:26:bb:15:8e:c5 src=c0:8c:60:00:ba:30 type=0x800 |<IP  id=43 frag=0 proto=tcp src=5.199.171.161 dst=193.50.11.32 |<TCP sport=http dport=45633 seq=785323 ack=0 flags=SA window=8192 |>`
Explications :

a. *les deux trames ont même adresses MAC source et destination ;*
b. *la première est une trame ARP de réponse de la machine source d'@MAC* `c0:8c:60:00:ba:30` *vers la machine cible d'@MAC* `00:26:bb:15:8e:c5` *:*
   - ◇ *les deux machines font partie du même réseau local ;*
   - ◇ *les machines sont configurée en IPv4, de la manière suivante :*

@MAC	*@IPv4*
c0:8c:60:00:ba:30	*193.50.10.254*
00:26:bb:15:8e:c5	*193.50.11.32*

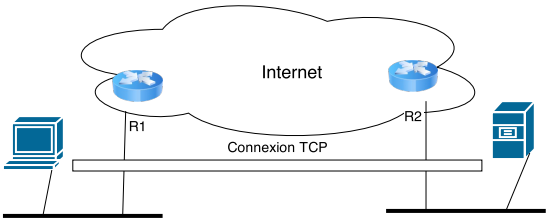
   - ◇ *le préfixe utilisé par le réseau local est au max un* `/23` *;*

c. *la seconde trame contient un datagramme IP qui provient de la même machine source que pour la trame 1, mais avec une @IP d'origine différente* `5.199.171.161` *:*
   - ◇ *la machine source d'@MAC* `c0:8c:60:00:ba:30`*, doit être un routeur ;*

   elle contient un segment TCP de port source 80, et des drapeaux `SA` *:*
   - ◇ *c'est une autorisation de connexion réalisée par la machine source d'@IP* `193.50.11.32` *;*
   - ◇ *la machine d'@IP* `193.50.11.32` *a réalisé une remande de connexion réussie vers le serveur Web hébergé sur la machine* `5.199.171.161`.

Programmation Socket – 5 points

5 – 5pts On aimerait pouvoir utiliser le logiciel de « *chat* » en UDP multicast entre les utilisateurs de deux réseaux interconnectés par Internet.
Une méthode consiste à écrire un logiciel qui servira de passerelle aux messages d'un réseau vers l'autre au travers d'une connexion TCP.



Questions :

a. Pourquoi n'est-il pas possible d'utiliser le logiciel de « *chat* » en UDP multicast à travers Internet ?
Parce que les datagrammes envoyés en multicast ne peuvent traverser Internet. Ils peuvent seulement traverser différents routeurs/switchs dans une même organisation à l'aide du protocole IGMP

b. Est-ce qu'une seule version du logiciel suffit ou faut-il deux versions et pourquoi ?
Le logiciel utilisant TCP, il faut un client et un serveur : il faut avoir deux versions du logiciel ou une version unique cumulant les deux rôles.

c. Décrivez comment un message envoyé dans un réseau en UDP multicast va pouvoir, grâce au(x) logiciel(s) d'atteindre les utilisateurs de l'autre réseau.
Le message va être reçu en UDP multicast par le logiciel client ou serveur, puis il va être transmis en TCP au travers d'Internet, pour être ensuite renvoyé en UDP multicast dans le réseau de l'interlocuteur.

d. Décrivez les opérations de la programmation Socket permettant de réaliser le logiciel réalisant ce travail.
Voici la description des opérations réalisée par le programme :
- ◇ *pour la partie TCP :*
  - ⋆ *Création d'une socket A pour le protocol TCP* `SOCKET, SOCK_STREAM` *;*
  - ⋆ *dans le cas du logiciel serveur, réalisation d'un* `bind` *sur la socket A vers le port choisi pour le protocol ;*
- ◇ *pour la partie UDP :*
  - ⋆ *Création d'une socket B pour le protocol UDP* `SOCKET, SOCK_UDP` *;*
  - ⋆ *Réalisation un* `bind` *sur la socket B vers l'adresse multicast et le port choisi pour le protocol ;*
  - ⋆ *Réalisation d'une opération pour joindre le groupe multicast sur la socket B* `SETSOCKOPT, IP_ADD_MEMBERSHIP` *;*
- ◇ *Travail du programme :*
  i. recevoir un message sur la socket UDP B `recvfrom` *:*
    - ▷ *analyser le message et extraire le contenu de ce message ;*
    - ▷ *envoyer le message sous forme d'une chaîne de caractères sur la connexion TCP de la socket A* `sendall` *;*

  ii. recevoir un message sur la socket TCP A :
    - ▷ *recevoir le message en tant que chaîne de caractères* `recv` *;*
    - ▷ *envoyer ce message en multicast sur la socket UDP B* `sendto`.

  iii.recommencer le travail en (i).