Université de Limoges | FACULTÉ DES SCIENCES ET TECHNIQUES

Master 1ère année

UE Réseaux Avancés I

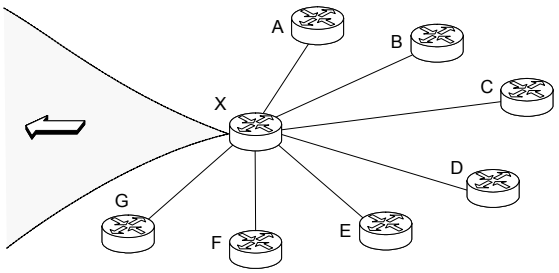
Corrections — novembre 2011

Durée : 2h — Documents autorisés

VLSM & CIDR — 6 points

1 – 3pts Soit la table de routage du routeur X utilisant le CIDR :

Subnet Mask	Next Hop
223.92.32.0 / 20	A
223.81.196.0 / 12	B
223.112.0.0 / 12	C
223.120.0.0 / 14	D
128.0.0.0 / 1	E
64.0.0.0 / 2	F
32.0.0.0 / 3	G



On va commencer par décomposer chacune des adresses fournies en binaire :

223.92.32.0/20	*11011111 01011100 0010*0000 00000000	A
223.81.196.0/12	*11011111 0101*0001 11000100 00000000	B
223.112.0.0/12	*11011111 0111*0000 00000000 00000000	C
223.120.0.0/14	*11011111 01111*000 00000000 00000000	D
128.0.0.0/1	*1*0000000 00000000 00000000 00000000	E
64.0.0.0/2	*01*000000 00000000 00000000 00000000	F
32.0.0.0/3	*001*00000 00000000 00000000 00000000	G

On remarque que l'adresse 223.81.196.0/12*, sortie* B*, correspond à celle d'une interface du routeur puisque des bits sont positionnés à droite de la limite du préfixe réseau.*
Pour déterminer le « next hop » ou « prochain saut », on recherche le plus long préfixe correspondant de la table de routage :

a. 195.145.34.2 $\Longrightarrow$ 11000001 00001100 01011001 00100010 $\longrightarrow$ sortie « E »
b. 223.95.19.135 $\Longrightarrow$ 11011111 01011111 00010011 10000111 $\longrightarrow$ sortie « B »
c. 223.95.34.9 $\Longrightarrow$ 11011111 01011111 00100010 00001001 $\longrightarrow$ sortie « B »
d. 63.67.145.18 $\Longrightarrow$ 00111111 01000011 10010001 00010010 $\longrightarrow$ sortie « G »
e. 223.123.59.47 $\Longrightarrow$ 11011111 01111011 00111011 00101111 $\longrightarrow$ sortie « D »
f. 223.125.49.47 $\Longrightarrow$ 11011111 01111101 00110001 00101111 $\longrightarrow$ sortie « C »

2 – 3pts Soit un routeur qui reçoit la liste suivante de datagrammes et qui les « route », de la façon suivante :

128.6.4.2	10000000 00000110 00000100 00000010	$\longrightarrow$ A
128.6.236.16	10000000 00000110 11101100 00010000	$\longrightarrow$ B
128.6.29.131	10000000 00000110 00011101 10000011	$\longrightarrow$ C
128.6.228.43	10000000 00000110 11100100 00001010	$\longrightarrow$ D

En les triant suivant les valeurs du 3ème octet (celui ou commence les différences) :

128.6.4.2	10000000 00000110 00000100 00000010	$\longrightarrow$ A
128.6.29.131	10000000 00000110 00011101 10000011	$\longrightarrow$ C
128.6.228.43	10000000 00000110 11100100 00001010	$\longrightarrow$ D
128.6.236.16	10000000 00000110 11101100 00010000	$\longrightarrow$ B

Les routes de sortie sont toutes différentes donc il faut que les préfixes soient tous différents.
On remarque :

- ▷ *pour que les deux premières routes du tableau trié, un préfixe* `/20` *est nécessaire pour les différencier ;*
- ▷ *pour les deux dernières, un préfixe* `\21` *est nécessaire ;*
- ▷ *suivant le préfixe* `\21` *les deux premières routes sont différentes des deux dernières.*

Ce qui donne la table de routage suivante :

réseau	*next hop*
`128.6.0.0/20`	*A*
`128.6.16.0/20`	*C*
`128.6.224.0/21`	*D*
`128.6.232.0/21`	*B*

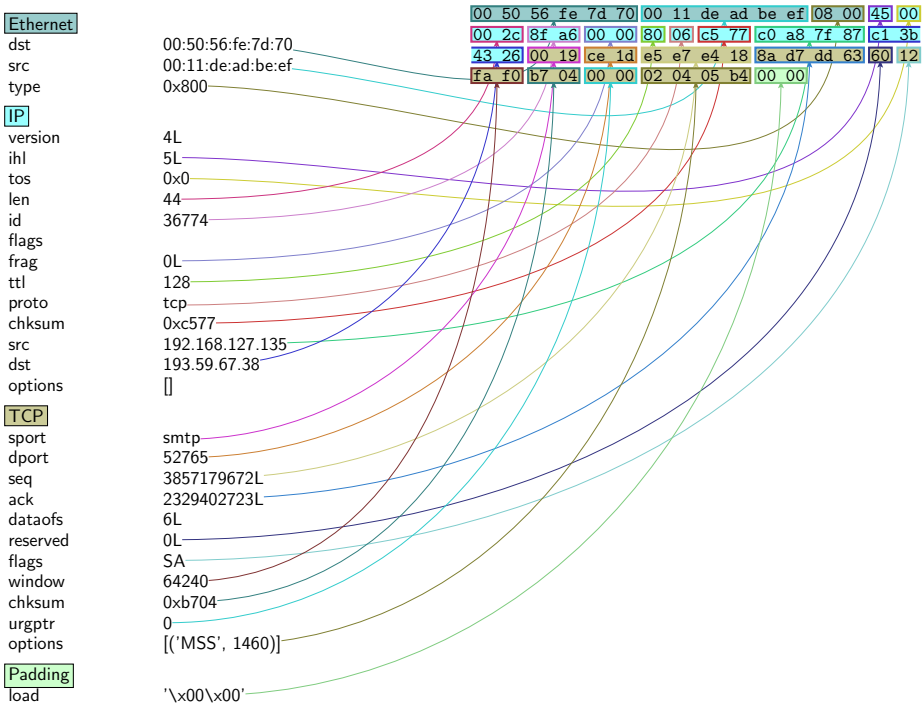
Datagramme IP & Analyse de trame — 7 points

3 – 3pts On a capturé cette trame :

```
0000   00 50 56 FE 7D 70 00 11  DE AD BE EF 08 00 45 00   .PV.}p........E.
0010   00 2C 8F A6 00 00 80 06  C5 77 C0 A8 7F 87 C1 3B   .,.......w.....;
0020   43 26 00 19 CE 1D E5 E7  E4 18 8A D7 DD 63 60 12   C&...........c`.
0030   FA F0 B7 04 00 00 02 04  05 B4 00 00
```

L'analyse de cette trame donne :

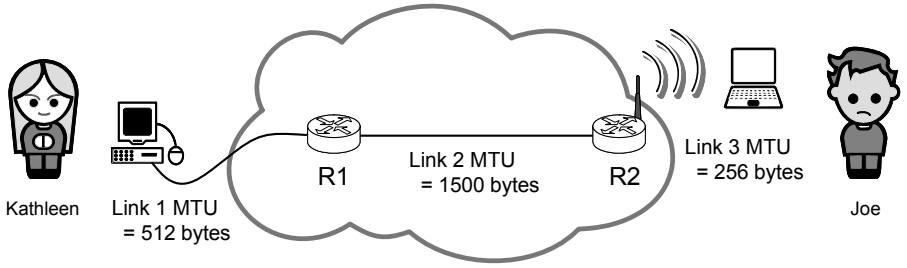
```
<Ether  dst=00:50:56:fe:7d:70 src=00:11:de:ad:be:ef type=0x800
|<IP  version=4L ihl=5L tos=0x0 len=44 id=36774 flags= frag=0L ttl=128 proto=tcp
chksum=0xc577 src=192.168.127.135 dst=193.59.67.38 options=[]
|<TCP  sport=smtp dport=52765 seq=3857179672L ack=2329402723L dataofs=6L reserved=0L
flags=SA window=64240 chksum=0xb704 urgptr=0 options=[('MSS', 1460)]
|<Padding  load='\x00\x00' |>>>>
```



On peut en déduire :

- ⋆ *c'est un segment TCP de réponse à une connexion vers le port associé au protocole SMTP ;*
- ⋆ *l'@IP source est dans un réseau privé,* `192.168.127.135`*, et l'@IP destination est* `193.59.67.38` *qui est une adresse globale ;*
- ⋆ *le TTL est de 128, pourtant il devrait être inférieur ⟹ manipulation par un « firewall » ;*
- ⋆ *le paquet partant sur Internet à partir d'un réseau privé ⟹, il y a un firewall réalisant du SNAT ;*
- ⋆ *le paquet étant allé d'Internet vers le réseau privé ⟹ le firewall a fait du DNAT ou du « port forwarding ».*

4 – Questions sur la fragmentation du datagramme IP :
4pts



a. Combien de datagrammes sont générés par l'ordinateur de Kathleen dans la couche IP ?
Chaque trame sur « Link 1 » fait au plus $512 octets$ *ce qui permet de transporter* $512-(30+20+20) = 442 octets$ *de données.*
Kathleen envoie $8*1024 = 8192 octets$ *de données, ce qui donne après division entière par* 442 *:* $8192 = 18*442+236$
Ce qui fait que Kathleen transmets 19 *datagrammes (18 de* $482 octets$ *et un de* $276 octets$*, chaucun avec* $40 octets$ *d'entête IP&TCP). On ne tient pas compte des segments de connexion et de fin.*

b. Combien de fragments Joe reçoit dans la couche IP ? Expliquez votre calcul.
Le MTU du « Link 3 » est de taille inférieure à celui du « Link 1 », il est donc nécessaire de fragmenter les paquets provenant de la machine de Kathleen :
- ◇ *sur le « Link 3 », les trames sont de taille* $256 octets$*, soit* $256-30 = 226 octets$ *pour le datagramme, et* $206 octets$ *pour le segment TCP ;*
- ◇ *les entêtes entre « Link 1 » et « Link 3 » sont de même taille :*
  - ⋆ *pour un datagramme de* $482 octets$*, il est fragmenté en* $462 = 2*200+62$ *(seulement* $200 octets$ *pour être un multiple de 8, conformément aux exigences de taille de fragment), soient 3 datagrammes : le premier et le second font* $200+20 = 220 octets$ *et le troisième fait* $62+20 = 82 octets$ *;*
  - ⋆ *pour un datagramme transportant un segment de* $256 octets$*, il est fragmenté en* $256 = 200+56$*, soient 2 datagrammes : un de* $200+20 = 220 octets$ *et le second de* $56+20 = 76 octets$*.*
- ◇ *en tout la fragmentation va créer :* $18*3+1*2 = 56$ *paquets.*

c. Décrivez les en-têtes des 4 premiers datagrammes et du dernier datagramme que Joe reçoit en indiquant **uniquement** la valeur des paramètres suivants : taille du datagramme, identifiant, offset et drapeaux (on choisira la valeur 672 comme identifiant du premier datagramme émis par Kathleen).

taille	identifiant	offset	drapeaux
220	672	0	MF
220	672	200	MF
82	672	400	-
220	673	0	MF
...	...	...	...
76	690	200	-

Les 4 premiers datagrammes reçus par Joe sont les 3 premiers fragments du premier datagramme envoyé par Kathleen, plus le premier fragment du second datagramme envoyé.

Le dernier datagramme reçu est le second fragment résultant de la fragmentation du dernier datagramme envoyé (numéro 19 soit l'identifiant 690).

d. Que se passe-t-il si le dernier datagramme se perd sur le lien 3 ? Combien de datagrammes IP seront retransmis par l'ordinateur de Kathleen ? Combien de fragments retransmis Joe va recevoir ?
Si le dernier datagramme se perd, alors le datagramme fragmenté auquel il appartient ne peut être reconstruit et il est entièrement annulé (tous les fragments reçus sont détruits).
L'ordinateur de Kathleen va retransmettre le dernier datagramme, n°19, car il n'aura pas eu confirmation de sa réception, ce qui donnera lieu à la réception de deux datagrammes de fragmentation sur la machine de Joe.

TCP – (3 points)

5 – 3pts Soient deux machine A et B qui accèdent à Internet par l'intermédiaire d'un routeur :

- ⋆ la machine A possède une fenêtre de réception configurée à $22ko$;
- ⋆ la machine B possède une fenêtre de réception configurée à $74ko$;
- ⋆ le réseau extérieur auquel est connecté le routeur pour aller sur Internet a une capacité de $10Mbps$.

Questions :

a. Si le RTT observé de A vers ce serveur est de $30ms$ avec quel débit maximum A va recevoir les données ?
La formule est débit $\leq \frac{taille\_fenêtre}{RTT}$*, soit débit* $\leq \frac{22*1024*8}{30.10^{-3}} \simeq 6Mbps$

b. Si le RTT observé de B vers ce serveur est de $50ms$ avec quel débit maximum B va recevoir les données ? *La formule est débit* $\leq \frac{74*1024*8}{50.10^{-3}} \simeq 12Mbps$ *mais le routeur possède un débit entrant d'au plus* $10Mbps$*, soit pour la machine B au final un débit d'au plus* $10Mbps$

c. Si les deux communications ont lieu simultanément quel débit vont-elles atteindre ?
Avec le contrôle de congestion intégré à TCP et s'il n'y a que deux communications simultanées alors le débit s'équilibrera à $5Mbps$ *pour A et B.*

Programmation Socket – (4 points)

6 – 4pts L'intranet de votre société est derrière un firewall qui n'autorise pas le passage des datagrammes UDP. L'administrateur système a imaginé la solution suivante :

- ▷ il existe une machine A connectée à Internet au delà du firewall qui héberge le site Web de la société ;
- ▷ une machine B de l'intranet qui veut envoyer un datagramme UDP va procéder de la manière suivante :
  - ◇ elle se connecte en TCP à la machine A au serveur « Redirection UDP » ;
  - ◇ elle envoie les données du datagramme UDP qu'elle veut envoyer ;
  - ◇ la machine A envoi alors le datagramme UDP.

L'administrateur vous demande d'écrire le programme du serveur TCP « Redirection UDP ».

a. Quelles sont les informations que devra transmettre la machine B à la machine A ?
Il faut transmettre le contenu du datagramme UDP, ainsi que le TSAP vers lequel l'envoyer.

b. Décrire les opérations de la programmation Socket permettant de réaliser le serveur « Redirection UDP ».
Il faut que le programme se comporte comme un serveur TCP :
- ◇ *il crée une socket,* `socket_tcp`*, en mode* `SOCK_STREAM`*;*
- ◇ *il attache la socket à un port* `socket_tcp.bind(('', port_serveur))`*;*
- ◇ *il configure la file d'attente* `socket_tcp.listen(socket.SOMAXCONN)`*;*
- ◇ *il exécute en boucle :*
  - ⋆ *il attend la connexion d'un client sur cette socket* `socket_tcp.accept()`*;*
  - ⋆ *lorsqu'un connexion se produit : il récupère les données (contenu et TSAP) :*
    - ▷ *il crée une socket,* `socket_udp`*, en mode* `SOCK_DGRAM`*;*
    - ▷ *il envoie le contenu vers le TSAP* `socket_udp.sendto(contenu, tsap_destination)`
  - ⋆ *il ferme la connxion établie avec le client* `socket_tcp.close()`.