Université de Limoges | FACULTÉ DES SCIENCES ET TECHNIQUES

Master 1ère année

UE Réseaux Avancés I

Corrections — janvier 2011

Durée : 1h45 — Documents autorisés

IPv6 (4 points)

1 – 4pts

a. Comment peut-il obtenir une adresse IPv6 de manière automatique ?
   - ◇ valable dans le réseau local :
     - ★ *par configuration « stateless » automatique, à partir de sa propre @MAC ou d'une adresse choisie de manière aléatoire, dont l'unicité sur le lien sera vérifiée à l'aide du NDP, « Neighbor Discovery Protocol » ;*
   - ◇ permettant de sortir du réseau local :
     - ★ *par configuration « stateful » à l'aide du protocole DHCPv6 ;*

b. Son adresse IPv6 « unicast de lien » : `FE80::020A:DEFF:FE00:326F/10`

c. **L'en-tête d'une trame** contenant un datagramme IPv6 en provenance du laptop et à destination de la machine d'@MAC `00:40:8D:05:00:34` :

```
00 40 8D 05 00 34 00 0A DE 00 32 6F 86 DD
```

d. On peut joindre ce laptop par son adresse IPv6 sans connaitre son @MAC : `33:33:FF:00:32:6F`

Programmation « Socket » (6 points)

2 – 6pts Votre projet de « robot-sortant-tout-seul-d'un-labyrinthe » a intéressé une entreprise de jouet fabriquant des robots télécommandés.

a. Comment communiquer avec le robot ?
Quels sont les avantages/inconvénients à utiliser TCP ou UDP ?
Les ordres donnés au robot sont très courts (au plus, une dizaine de caractères), ce qui tient dans le contenu d'un seul datagramme.
Le protocole TCP « consomme » 3 datagrammes uniquement pour établir la connexion puis au moins deux datagrammes supplémentaires pour mettre fin à la connexion.
TCP permet de facilement renvoyer un message de confirmation de la part du serveur embarqué dans le robot ou de bénéficier d'un contrôle automatique de cet envoi (pas de problème avec la communication de son établissement à sa terminaison).
Le protocole UDP peut suffire à l'envoi de la commande en un seul datagramme. Par contre il est nécessaire de gérer l'accusé de réception de la commande par l'envoi d'un datagramme de réponse de la part du serveur.
En dernier point, on peut considérer que TCP l'emporte, lorsque le même client envoie des ordres successifs au robot au sein de la même connexion, puisque TCP garantie que les ordres arrivent dans le bon ordre et sans perte et que le surcoût évoqué plus haut est diminué sur la durée de l'échange de plusieurs ordres.

b. Si on veut s'assurer que chaque ordre a bien été reçu **une seule fois**, comment le faire en TCP et en UDP ?
Pour TCP, on peut envoyer sous forme de réponse de la part du serveur une confirmation de la réception de l'ordre ou s'assurer que la connexion TCP s'est terminée sans rupture de connexion.

Pour UDP, il faut renvoyer un datagramme UDP de confirmation ainsi qu'un mécanisme de réémission. Pour associer l'un à l'autre il faut les estampiller avec un numéro dont le client assure l'évolution (incrémentation à chaque ordre par exemple).

c. Donnez le **code Python du serveur** présent sur le Robot.
On va utiliser TCP et une seule connexion pour plusieurs ordres à la suite dans la même connexion.

```
import socket, ControlRobot

adresse_hote = ''
numero_port = 7777
socket_serveur = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
socket_serveur.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
socket_serveur.bind((adresse_hote, numero_port))
socket_serveur.listen(1) # un client à la fois

re_nord = re.compile(r'^tourner nord',re.I)
re_avancer = re.compile(r'^avancer', re.I)
# à compléter
exp_fin = re.compile(r'^fin', re.I)
while 1:
    (connexion, depuis) = socket_serveur.accept()
    connexion.sendall("Control Robot v1.0\n")
       while 1:
           ligne = connexion.recv(1000)
           if not ligne: break
               resultat = re_nord.search(ligne)
               if resultat :
                       tourner_nord()
                       connexion.sendall("OK Tourner nord\n")
                       continue
               resultat = re_avancer.search(ligne)
               if resultat :
                       avancer()
                       connexion.sendall("OK Avancer\n")
                       continue
               # à compléter
               resultat = re_fin.search(ligne)
               if resultat :
                       connexion.sendall("OK Fin\n")
                       break
    connexion.close()
ma_socket.close()
```

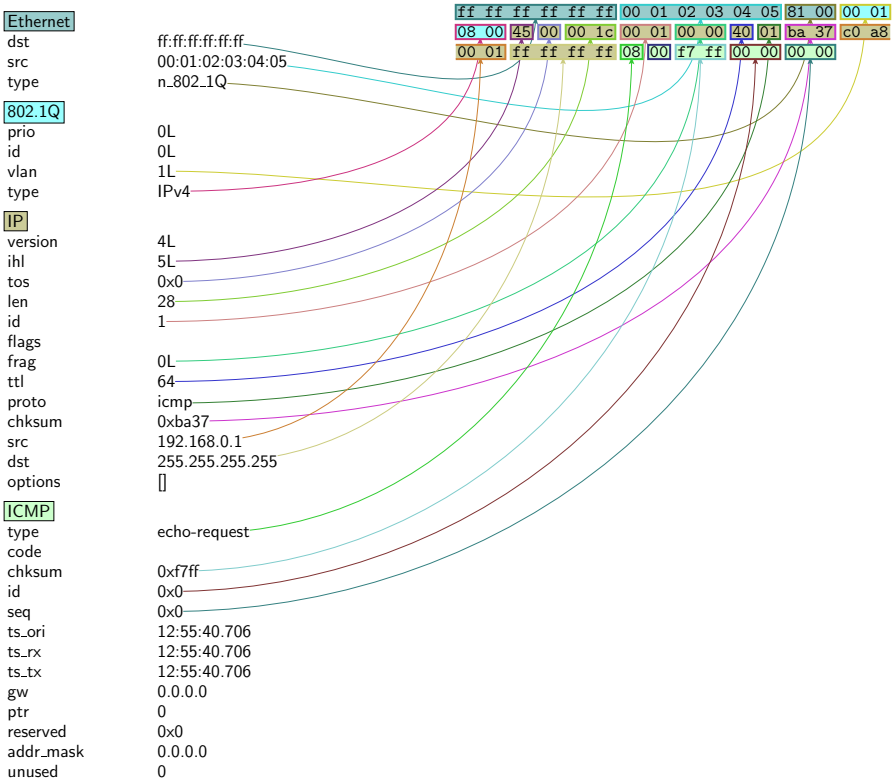
d. Comment **faciliter la configuration** du programme client, pour ne pas avoir à saisir l'@IP du serveur présent sur le robot (ce qui peut être compliqué pour un enfant) ?

Le serveur peut utiliser une diffusion dans le réseau local pour indiquer sa présence et sa configuration : broadcast en UDP sur une adresse de groupe associée au contrôle du robot et partagée entre le client et le serveur.
Ainsi, le client détecte quelle est l'adresse du serveur et peut se configurer automatiquement.
S'il existe plusieurs robots connectés au même réseau local, alors on peut utiliser l'adresse MAC pour les distinguer.

Audit réseau (3 points)

3 – L'analyse de la trame donne :

3pts



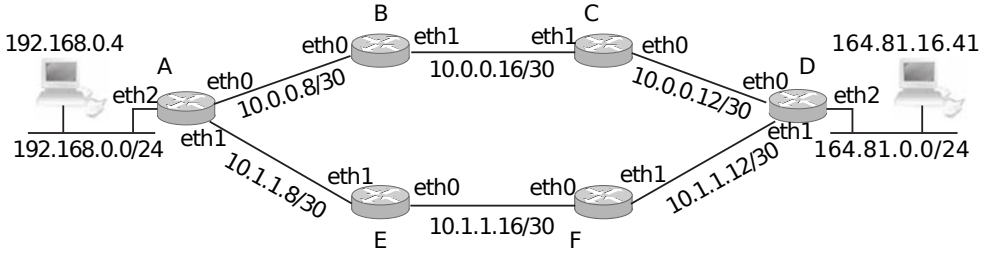
Entre quels matériels réseaux cette trame a-t-elle circulé ?
Le datagramme est encapsulé dans une trame de type VLAN, cette trame circule alors dans un Trunk entre deux switchs.

Firewall & QoS (7 points)

4 – D'après les connexions des différents routeurs :

7pts

a. Retrouvez le schéma du réseau d'interconnexion de ces différents routeurs :



b. On veut automatiser la procédure de configuration des tables de routages de ces différents routeurs, que conseillez vous entre OSPF et RIP ?
La taille minimale du réseau invite à sélectionner RIP, mais OSPF apporte l'avantage de pouvoir tirer parti des deux chemins possibles pour aller de `192.168.0.0/24` *à* `164.81.0.0/24`.

c. La politique de sécurité
   - ◇ empêcher les machines du réseau `192.168.0.0/24` de communiquer vers l'extérieur :
     - ★ Sur le Routeur A :
     - ★ si on veut bloquer la totalité des échanges :

```
# iptables -t filter -P FORWARD DROP
```

★ ou seulement ceux en provenance de 192.168.0.0/24 :

```
# iptables -t filter -A FORWARD -s 192.168.0.0/24 -j DROP
```

◇ les machines du réseau 192.168.0.0/24 sont autorisées à accéder à un serveur Web d'adresse 164.81.16.41 :

```
# iptables -A FORWARD  -m state --state ESTABLISHED -j ACCEPT
# iptables -t filter -A FORWARD -s 192.168.0.0/24 -d 164.81.16.41 -p tcp --dport 80
-m state --state NEW -j ACCEPT
```

Avec le choix du blocage sans utiliser la « policy », il faut s'assurer que ces deux règles sont placées avant celles de blocaqge (on peut utiliser l'insertion au lieu de l'ajout).

d. On veut empêcher l'accès SSH (port 22) depuis le réseau 164.81.0.0/24 sur le routeur D :

```
# iptables -t filter -A INPUT -s 164.81.0.0/24 -p tcp --dport 22 -j DROP
```

e. Soit la configuration suivante pour réaliser de la QoS :

```
# tc qdisc add dev $IF root handle 1: htb default 10
# tc class add dev $IF parent 1: classid 1:10 htb rate 20mbps
# tc class add dev $IF parent 1: classid 1:20 htb rate 10mbps
```

On veut limiter le trafic du protocole VNC à 10mbps :

Le protocole VNC est basé sur TCP (port 5900) et permet de visualiser l'écran d'un ordinateur à distance.

Sur quel routeur doit-on installer cette configuration de QoS, si l'accès par VNC se fait depuis une machine du réseau 192.168.0.0/24 vers une machine du réseau 164.81.0.0/24 (c-à-d. qu'une machine du réseau 164.81.0.0/24 envoi régulièrement le contenu de son écran à une machine du réseau 192.168.0.0/24) ?

Le trafic « volumineux » se fait d'une machine du réseau 164.81.0.0/24 *vers une machine du réseau* 192.168.0.0/24*, c'est donc lui qu'il faut limiter.*
Pour le limiter, il faut utiliser le routeur le plus proche de l'émetteur, à savoir le routeur D.

```
# iptables -t mangle -A PREROUTING -p tcp --sport 5900 -j MARK --set-mark 1
# tc filter add dev $IF protocol ip parent 1: handle 1 fw classid 1:20
```