Année Universitaire 2010-2011 session 1 d'automne

Parcours : CSB5 Licence 3 STS Informatique **UE :** INF157
Épreuve : Examen Utilisation des réseaux
Date : 13 décembre 2010 **Heure :** 8h30 **Durée :** 1h30
Modalités : Épreuve sans document, avec calculatrice
Épreuve de : Mr Samuel Thibault 5 pages

Exercice 1. Questions de cours (5 pt)

Q1.1 Faites un schéma pour représenter l'empilement habituel des protocoles Ethernet, IP, TCP et UDP.

TCP	*UDP*
IP	
Ethernet	

Q1.2 Donnez la première ligne d'une requête HTTP (version 1.1) demandant le document `bobi.txt` situé dans le répertoire `tmp` du serveur web `foo.com` interrogé. Donnez la première ligne de la réponse HTTP de celui-ci, lorsqu'il n'a pas trouvé le document demandé. (2 lignes max)

GET /tmp/bobi.txt HTTP/1.1

HTTP/1.1 404 Not Found

Q1.3 Pourquoi préfère-t-on pour le protocole ssh utiliser TCP plutôt que UDP, alors que pour DNS on se permet d'utiliser UDP ?

ssh est censé garantir l'envoi correct d'un ensemble éventuellement grand de données (e.g. pour un transfert scp, mais aussi pour éviter les fautes de frappes au prompt), il vaut donc mieux utiliser TCP qui gère les retransmissions nécessaires. Pour DNS, puisqu'une requête tient dans un seul datagramme UDP, un seul paquet suffit, il n'y a pas besoin de connexion, et s'il y a une perte, c'est la requête toute entière qui est perdue, il n'est pas difficile pour l'application de simplement réessayer.

Q1.4 À quoi sert une passerelle ?

Un routeur a typiquement des interfaces réseaux dans différents réseaux. Il s'occupe de retransmettre les paquets IPs d'un réseau à un autre. D'un point de vue des ordinateurs de ces réseaux, ce routeur sert donc de passerelle *vers les autres réseaux.*

Q1.5 Un correspondant au Japon m'a envoyé par mail une archive .zip contenant un document sur lequel il vient de travailler. Mon dézippeur affirme que le document vient du futur ! Que s'est-il passé, qu'aurait-il dû se passer ? (Il m'affirme que son ordinateur est bien à l'heure)

Le Japon a un fuseau horaire différent de la France : il y est "plus tard". La date (à l'heure japonaise) du document inscrite dans l'archive est donc numériquement ultérieure à la date actuelle (à l'heure française). Le problème est que le dézippeur n'est pas au courant de ce décalage horaire. Il faudrait enregistrer les dates dans l'archive soit systématiquement à l'heure UTC, soit avec une indication de fuseau horaire, pour que le dézippeur puisse calculer le décalage automatiquement.

Q1.6 Quels sont tous les protocoles et machines potentiellement mises en œuvre lorsque l'on lance la commande `ping -c 1 www.tfou.fr` ? Énumérez les paquets émis et reçus par la carte réseau de la machine où l'on lance la commande (on supposera que la machine vient juste d'être allumée).

ping tente d'abord de résoudre le nom www.tfou.fr *via le protocole DNS. Pour cela il faut envoyer un datagramme UDP via IP à l'adresse du serveur DNS. Puisque la machine vient d'être allumée, une requête ARP est d'abord nécessaire pour obtenir l'adresse Ethernet correspondant à l'adresse IP du serveur DNS s'il est sur le même réseau, ou bien à l'adresse de la passerelle sinon. La réponse DNS revient de la même façon.*

ping peut alors envoyer une requête ICMP echo request via IP, en faisant éventuellement une requête ARP pour obtenir l'adresse Ethernet correspondant à l'adresse IP de la passerelle, si cela n'a pas déjà été fait pour la requête DNS.

Q1.7 En quoi la version *threads* d'un serveur echo est-elle plus simple qu'une version `select` ?

La version threads *permet d'écrire l'ensemble de la discussion entre serveur et client au sein d'une simple boucle read/write ne gérant qu'une connection. Il n'y a pas besoin d'appeler select, puis d'itérer sur l'ensemble des sockets.*

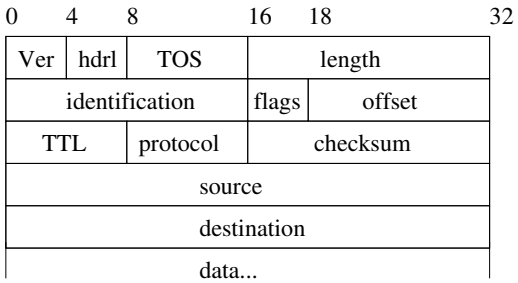
Cela simplifie par ailleurs le stockage des données : penser notamment au cas où un client fournirait beaucoup de données, mais n'en consommerait pas : le tampon d'émission de la socket se remplirait, rendant alors write() bloquant !.

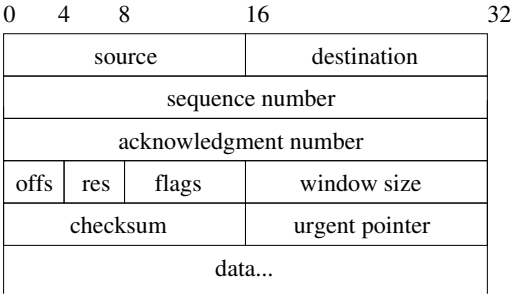
Exercice 2. Analyse de paquet

Voici un paquet IP capturé par *wireshark*, contenant un extrait de connexion ssh (dont on rappelle que le numéro de port est 22) :

```
0x00:  45 00 00 48 5d cb 40 00 3a 06 16 2f 0b 0c 0d 0e
0x10:  0b 0c 0d 0f 00 16 04 01 f7 90 50 b5 18 fa 80 3f
0x20:  80 18 00 2e 47 f2 00 00 01 01 08 0a 1c 92 0d 8a
0x30:  00 3a b7 ac 53 53 48 2d 32 2e 30 2d 4f 70 65 6e
0x40:  53 53 48 5f 34 2e 33 0a
```

On rappelle le format des en-têtes IP et TCP :





Quelle est l'adresse IP du serveur sshd, et l'adresse IP du client ssh ? Quel est le numéro de port utilisé du côté du client ?

Puisque c'est un paquet IP, on suit d'abord le format de l'en-tête IP, les octets correspondant aux adresse IP sont donc `0b 0c 0d 0e` *et* `0b 0c 0d 0f`*. L'adresse de l'émetteur est donc 11.12.13.14, et celle du récepteur, 11.12.13.15. A priori, comme d'habitude UDP est encapsulé directement dans les données du paquet IP, on saute donc les 5*32 bits d'en-tête IP, et l'on suit le format de l'en-tête UDP. Les octets du numéro de port source (côté émetteur) sont donc* `00 16` *donc 22, et les octets du numéro de port destination (côté récepteur) sont* `04 00`*, donc 1024. Puisqu'il s'agit donc d'un datagramme UDP en provenance du port 22 vers le port 1024, il s'agit d'un datagramme venant du serveur ssh (port 22) vers le client. L'adresse IP du client est donc celle du récepteur, 11.12.13.14, et le port de son côté est 1024.*

Exercice 3. Calculs de débits

Deux fous utilisent des clés USB de 32Go et des lance-pierres pour s'échanger des données. Le temps de vol d'une clé USB entre les deux fous est d'environ une seconde. Le temps de préparation du lance-pierre (pose, visée, tir) est d'environ deux secondes, et le temps de réception est d'environ une seconde.

Q3.1 Dans un premier temps, on considère qu'ils ont tous deux des clés USB contenant déjà les données prêtes à envoyer, et qu'ils n'ont pas besoin de lire celles qu'ils reçoivent immédiatement. Quels sont le débit et la latence de ce moyen de communication ? (expliquez votre calcul)

Un fou émetteur prépare et lance ses clés en 2 secondes, il peut donc tirer des clés toutes les 2 secondes, soit un débit de 32Go / 2s = 16Go/s.

Est-ce que le fou réceptionneur peut supporter un tel débit ? Oui, puisqu'il peut réceptionner une clé en une seule seconde.

La latence d'un aller-retour nécessite préparation, vol et réception de la requête, puis préparation, vol et réception de la réponse, soit 8 secondes en tout.

Q3.2 Maintenant, un des deux fous veut envoyer à l'autre un gros fichier de son ordinateur portable, en le découpant en morceaux de 32Go. La vitesse de transfert entre l'ordinateur et la clé USB est de 10Mo/s. On négligera le temps de débranchement/rebranchement de clé USB. Quels sont le débit et la latence du transfert vers l'ordinateur de l'autre fou ? (expliquez votre calcul)

Puisque maintenant il faut transférer les données sur l'ordinateur via USB, le débit d'ensemble est pénalisé par ce goulet d'étranglement, et donc réduit à 10Mo/s : au mieux, les clés USB s'entassent dans une corbeille de réception en attendant d'être insérées (en temps négligeable) pour transfert sur l'ordinateur.

La latence comporte désormais le temps de transfert vers l'ordinateur. À 10Mo/s, il faut 32Go / 10Mo/s = 3200 secondes (presqu'une heure !) pour transférer les 32Go d'une clé. Le temps de préparation, de vol et de réception deviennent négligeables.

Q3.3 Le fou émetteur vise en fait assez mal, et un certain nombre de clés USB sont ainsi perdues en chemin (heureusement, ils en ont en réserve). En vous inspirant d'un protocole bien connu, expliquez brièvement comment les deux fous peuvent s'assurer que le fou récepteur reçoit correctement le fichier.

On peut bien sûr s'inspirer de TCP : il suffit que le fou émetteur numérote les clés, le fou récepteur peut alors facilement vérifier s'il lui en manque, et envoyer au récepteur une clé sur laquelle il a noté quelle(s) clé(s) lui manque, pour qu'il les lui réenvoie de nouveau.

Exercice 4. Voici la configuration d'une machine :

```
$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:21:70:b4:36:49
          inet adr:169.254.255.8  Bcast:169.254.255.255  Masque:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:273345 errors:0 dropped:0 overruns:0 frame:0
          TX packets:123007 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          RX bytes:86514668 (82.5 MiB)  TX bytes:23180492 (22.1 MiB)
$ /sbin/route
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
169.254.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         169.254.0.1     0.0.0.0         UG    0      0        0 eth0
$ /usr/sbin/arp
Address                  HWtype  HWaddress           Flags Mask            Iface
```

Q4.1 Quelle sont l'adresse MAC et l'adresse IP de la machine ?

MAC : 00 :21 :70 :b4 :36 :49

IP : 169.254.255.8

Q4.2 Quel est l'ensemble des adresses IP accessibles directement ?

Le masque réseau étant 255.255.0.0, l'ensemble des adresses accessible va de 169.254.0.1 (c'est d'ailleurs la passerelle) à 169.254.255.254. Il y a de plus l'adresse de broadcast 169.254.255.255.

Exercice 5. On désire implémenter un jeu d'échecs en ligne permettant d'obtenir une liste d'adversaires potentiels, en choisir un et jouer contre lui. On réfléchit au protocole que l'on va inventer pour cela.

Q5.1 On a vu en cours qu'un protocole peut être centralisé, décentralisé, ou bien acentré. Qu'est-ce que cela voudrait dire en pratique dans ce contexte de jeu d'échec en ligne ?

Un jeu d'échec en ligne centralisé comporterait un seul serveur central auquel tous les clients doivent se connecter pour pouvoir jouer ensemble.

Un jeu d'échec décentralisé comporterait une hiérarchie de serveurs, les listes de clients et demandes de parties étant propagées le long de la hiérarchie.

Un jeu d'échec acentré comporterait différents serveurs, sans hiérarchie (et donc sans centre), s'échangeant les listes de clients et demandes de parties à l'aide d'un protocole peer-to-peer.

Q5.2 On se propose d'écrire un client que les utilisateurs lanceront sur leur machine. Quelles informations fera-t-on passer sur le réseau ?

On peut faire tourner l'essentiel du jeu sur la machine client, il ne reste alors qu'à propager sur le réseau les demandes de parties et les coups joués.

Q5.3 Le service marketting nous indique qu'il faudrait que le jeu puisse fonctionner depuis un simple navigateur web mais en conservant un rendu 3D de l'échiquier. En supposant que les navigateurs webs disposent de l'extension javascript ou flash, quelles sont toutes les informations que l'on fera passer sur le réseau ?

Il faudra cette fois en plus envoyer l'application javascript ou flash, comportant notamment les rendus visuels des pièces.

Q5.4 Le service clientèle indique que des utilisateurs se sont plaints de ne pas pouvoir jouer sans extension javascript ni flash. Quelle version simplifiée peut-on fournir ?

On peut écrire une version basée purement sur des pages html, comportant par exemple une table d'images pour montrer l'échiquier, et des boutons de formulaire CGI pour soumettre les coups.

Exercice 6. PGP

Le système cryptographique PGP est basé sur des paires de clés publique/privée. Toute personne désirant utiliser PGP génère une paire de clés publique/privée, garde la partie privée secrète, et publie la partie publique le plus largement possible en la mettant à disposition sur des serveurs publics bien connus, sur lesquels n'importe qui peut déposer des clés publiques à volonté, et où n'importe qui peut les récupérer.

Q6.1 Alice a généré une paire de clés et Bob en a récupéré la partie publique. Ils peuvent alors utiliser les clés dont ils disposent ainsi pour protéger plus ou moins les messages qu'ils échangent. Qui peut envoyer des message cryptés à qui ? Qui peut envoyer des messages signés à qui ?

Puisque Bob a la partie publique d'Alice, il peut vérifier qu'Alice est bien émettrice des messages qu'elle signe avec sa clé privée, puisqu'elle seule a cette clé. Alice peut certes crypter ses messages avec sa clé privée, mais n'importe qui pouvant récupérer la partie publique, cela n'a guère d'intérêt, puisque tout le monde peut donc les lire. Inversement, Bob peut certes signer un message avec la clé publique d'Alice, mais n'importe qui pourrait le faire, ça n'a donc pas d'intérêt non plus. Il peut par contre crypter un message, et seule Alice, grâce à la clé privée, peut le lire, ce qui est intéressant.

Q6.2 Il n'est pas très fiable de déposer simplement une clé publique sur un serveur public ; que pourrait faire quelqu'un mal intentionné ?

Il pourrait déposer une clé publique sous le nom de quelqu'un d'autre, faisant ainsi croire que c'est sa clé, et signer alors des messages sous son identité et se faire envoyer les messages cryptés !

Le plus sûr est de se rencontrer physiquement pour pouvoir s'échanger en main propre des clés publiques, mais ce n'est pas toujours possible. C'est pour cela qu'en fait, un utilisateur dépose sur les serveurs publics

une version de sa clé publique *signée* par d'autres personnes, dont il suffit ainsi d'avoir la clé publique pour pouvoir être sûr que la clé signée est bien digne de confiance.

Q6.3 Bob génère aussi une paire de clés. Ni Bob ni Alice n'ont le temps de se déplacer pour s'échanger leurs clés publiques en mains propres. Cependant, un ami commun Joe, aussi utilisateur de PGP, va bientôt déménager : il pourra rencontrer physiquement Alice puis Bob (mais ne reviendra pas voir Alice). Expliquez comment procéder pour que Bob et Alice puissent échanger leurs clés publiques de manière sûre.

Joe donne en main propre sa clé publique à Alice et récupère la clé publique d'Alice. L'un et l'autre sont donc bien sûrs d'avoir les clés d'autrui. Joe va ensuite voir Bob, et lui donne en main propre la clé publique d'Alice. Puisque Joe est un ami, Bob a confiance en lui, et est donc bien sûr d'avoir la clé publique d'Alice. Bob donne sa clé publique à Joe. Joe la signe avec sa clé privée, et envoie le résultat à Alice. Alice peut vérifier, à l'aide de la clé publique de Joe, que c'est bien Joe qui a signé cette clé. Puisqu'elle fait confiance à son ami Joe, elle est sûre que c'est bien la clé de Bob qu'elle a ainsi reçu.