M1 - Outils de l'Internet (Année 2010/2011)

Adressage et nommage sur l'Internet

L'objet de ce premier TD est de rappeler les notions fondamentales nécessaires à l'écriture d'applications réseau sur des réseaux TCP/IP avec serveurs DNS en utilisant une API de type sockets BSD (ici *à la Python*).

Exercice 1. *Adresses IP et sockets TCP*

Une socket TCP est un canal de communication bidirectionnel qui abstrait la communication connectée par paquets en un flux de donnée : tout se passe comme si l'on communiquait à travers un fichier ou un pipe. Du point de vue réseau une socket est un quadruplet constitué des adresses IPs des deux extrémités (x.y.z.t) et des ports TCP utilisés pour réaliser cet échange. La syntaxe de l'API python pour les sockets est très similaire à l'API C BSD standard (à la notation objet pointée près).

1. Rappeler la structure d'un serveur synchrone qui traite séquentiellement les connexions reçues jusqu'à leur fin avant de passer à la suivante. Illustrer par un pseudo-code de serveur echo.

2. Quelles modifications faut-il apporter pour traiter plusieurs connexions en parallèle ? Proposer une solution à l'aide de la fonction `select` (dont la signature est de la forme `a,b,c=select.select(ap,bp,cp,60)`).

3. Qu'apportent dans ce contexte les threads ou forks ? Quels sont les inconvénients ? Illustrer à l'aide d'un exemple utilisant des processus légers.

Exercice 2. *Tunnel TCP, NAT, VPN*

1. Un tunnel TCP est un serveur qui transmet toute information à travers une connexion reçue sur le port n de la machine A vers le port m de la machine B. Expliquer comment mettre en oeuvre ce principe avec des sockets.

2. Un NAT est un système de traduction d'adresses, généralement mis en oeuvre sur un routeur. Toute connexion de l'intérieur du domaine vers l'extérieur est émise à travers le NAT en utilisant l'adresse IP de ce dernier. Proposer une explication de fonctionnement, détailler la gestion des connexions TCP par un NAT (établissement de la connexion, transmission des données, etc).

3. Dessinez le réseau suivant : un serveur Web (`192.168.0.53`) sur le port 80 est sur un sous-réseau NAT (`192.168.0.x`) représenté par l'IP `10.0.0.32` dans un autre sous-réseau NAT (`10.0.0.x`) lui-même ayant l'IP publique `82.216.35.19`. Que se passe-t-il lorsque M. Doe (`140.97.32.21`) essaye de se connecter au serveur Web ?

Exercice 3. *Nommage sur l'internet : URI et DNS*

1. Représenter graphiquement les différentes requêtes envoyées par un solveur DNS à partir d'un cache vide pour résoudre l'adresse `www.cri.up.univ-mrs.fr` (on supposera que tous les sous-domaines jusqu'à `cri` disposent d'un serveur de nom propre).

2. Que se passe-t-il si subséquemment il recharge la même page ? Et s'il décide de consulter `www.pysique.up.univ-mrs.fr` ? Puis `www.univ-mrs.fr` ?

3. Le DNS inverse permet d'associer à toute adresse IP un nom canonique. Quelles sont les modifications à apporter au DNS pour permettre de retrouver un nom à partir de l'adresse IP en invoquant pour l'adresse `x.y.z.t` le DNS sur `t.z.y.x.in-addr.arpa` ? Sur quels serveurs les informations sont-elles stockées ?

Exercice 4. *The big picture*

1. Quelles étapes successives (DNS, connexion via socket) faut-il accomplir pour passer d'une URI type `http://lapin.org:8987/blabla/bli/blu` à un échange de deux processus à travers une connexion TCP suivant un protocole fixé ?

2. Comment le protocole HTTP permet-il à une même machine possédant une seule adresse IP de servir plusieurs sites Web différents suivant le nom avec lequel on l'appelle ?

3. Proposer plusieurs techniques (s'appuyant sur des niveaux différents : IP, TCP, DNS, etc) pour un gros fournisseur de service (par exemple `http://update.microsoft.com`) de multiplier les serveurs de manière transparente ?