TELECOM nancy

UNIVERSITÉ DE LORRAINE

ESIAL 2A - RSA -Réseaux

2015-2016

TP Programmation Avancée

L'ensemble des sources dont vous avez besoin pour votre TP sont à votre disposition à l'URL suivante : `http://www.loria.fr/~ichris/Templates.tar.gz`. La répertoire `Templates` est composé de 4 sous-répertoires :

- Select
- Raw
- Urgent
- IPv4

Le TP se déroulera en deux séances. A la fin de chaque séance, vous enverrez à votre encadrant les exercices que vous avez finis de réaliser et vous supprimerez localement les fichiers que vous avez écrits.
Si au bout des deux séances, vous n'avez pas implanté l'ensemble des exercices, vous avez jusqu'au **Mardi 13 mai 2014** pour envoyer le tout à votre encadrant.

Exercice 1

Dans le répertoire **Raw** se trouvent les sources d'un client UDP qui envoie l'heure à un serveur, attend un message du serveur et quitte l'application. Développer le serveur qui affiche l'heure reçue, renvoie la donnée reçue et se met ensuite en attente sur la requête d'un autre client.

Exercice 2

L'objectif de cet exercice est de vous faire manipuler le concept d'entrées/sorties multiplexées.

- Tester tout d'abord la version du serveur echo multi-client : **Select/servmulti_tcp.c**. Pour la partie client, vous pouvez utiliser **telnet**.
- Modifier cette version pour en faire un serveur multi-clients utilisant la primitive **select()**. Il s'agit d'implanter les notions vues en TD.

Exercice 3

L'objectif de cet exercice est de vous faire manipuler le concept de socket raw.

- Tout d'abord copier le fichier **Raw/icmpd-incomplet.c** dans **icmpd.c** et complétez-le
- Tester les différents scénarios permettant l'arrivée de messages ICMP de type ICMP_ECHOREPLY, ICMP_UNREACH et ICMP_TIMXCEED. Vous pourrez également vérifier la réception des messages ICMP via **wireshark**.

Exercice 4

L'objectif de cet exercice est de vous faire manipuler le concept de donnée urgent appelée aussi message OOB (Out Of Band).

- Tout d'abord copier le fichier **Urgent/serverUrgentTCP-incomplet.c** dans **serverUrgentTCP.c** et complétez-le.
- Exécuter le serveur en connectant le client dont le fichier source est **Urgent/clientUrgentTCP.c**. Analyser et expliquer le comportement du serveur.
- Observer avec **wireshark** l'en-tête TCP et notamment les informations relatives aux données urgentes.

Exercice 5

L'objectif de cet exercice est de vous faire manipuler le concept de multicast/communication de groupe.

- Ecrire le code de l'émetteur `sender.c` qui enverra la date courante à une adresse multicast. Il suffit pour cela d'adapter le code source **Raw/clientUDP.c**.
- Ecrire le code du récepteur qui affiche la date courante reçue sur cette adresse multicast.
- Tester votre application en lançant plusieurs récepteurs sur une même machine (voire sur d'autres machines).

Exercice 6

L'objectif de cet exercice est de vous faire manipuler les adresses IPv6.

- Porter en IPv6 le code relatif aux sockets TCP/IPv4 qui se trouve dans le sous-répertoire IPv4. Pour vous aider, des informations sont également disponibles le site de `http://livre.g6.asso.fr/`, dans la section Programmation d'applications. Pour le portage, deux versions sont demandées :
  1. une première version qui prend en compte la modification des structures de données des adresses.
  2. une deuxième version avec utilisation de l'appel getaddrinfo. Un exemple de client-serveur est donné sur le site. N'hésitez pas non plus à faire un **man de getaddrinfo.**
- Tester votre code sur le réseau local. Vérifier que la communication se fait bien en IPv6.

Exercice 7

Dans la cadre du TP/Projet, vous devez réaliser un système d'échange de fichiers en mode pair à pair (P2P). Le système est composé :
(1) d'un serveur central qui stocke les métadonnées des fichiers à échanger et met en relation les pairs ;
(2) d'un pair qui agit à la fois comme client et serveur. Il doit pouvoir réaliser les fonctionnalités de publication, de recherche et d'échange (cf. Figure 1)

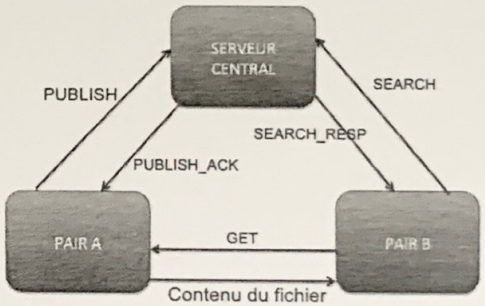

FIGURE 1 – Architecture globale

Publication. Pour chaque fichier qu'il souhaite mettre à disposition, le pair envoie vers un serveur central une requête PUBLISH dans laquelle il spécifie les métatdonnées concernant ce fichier à savoir :

- le nom du fichier ;
- le type de fichier (mp3, jpeg, txt,...) ;
- une liste de mots clés décrivant son contenu ;
- le hash du contenu. Vous utiliserez la fonction de hachage SHA-1.

Le pair stocke localement les fichiers à publier dans un même répertoire.

Le serveur central acquitte le demande de publication via un message PUBLISH_ACK. En cas de problèmes, un message d'erreur est retourné au pair.

Recherche. Le pair interroge via une requête SEARCH le serveur central pour connaitre la liste des sources (pairs) disposant des fichiers correspondant à un mot-clé fourni par l'utilisateur. Le serveur central retourne le résultat de sa recherche dans une réponse SEARCH_RESP où il donne pour les fichiers obtenus :

- l'adresse IP de la source ;
- le nom du fichier ;
- le type de fichier ;
- la liste des autres mots-clés ;
- le hash du contenu.

Le serveur central doit donc pouvoir effectuer une recherche à partir d'un mot-clé.

Echange. Si l'utilisateur sélectionne une des sources en fonction des informations retournées, le pair contacte la source associée afin de récupérer le fichier demandé. Il est possible que le pair distant ne soit plus connecté ou qu'il ne dispose plus du fichier. Vous devrez gérer ces cas d'erreur et proposer à l'utilisateur d'effectuer une autre requête. Une fois le fichier téléchargé, le pair vérifie que le hash de son contenu correspond à celui récupéré dans les metadonnées. Si tel n'est pas le cas, le fichier n'est pas conservé. Le pair stocke localement les fichiers téléchargés dans un même répertoire.

Le protocole UDP sera utilisé entre un pair et le seveur central et le protocole TCP entre les pairs pour les échanges de fichiers.
Vous livrerez le code source (format tar.gz) avec un Makefile permettant la génération des deux exécutables (celui du pair et celui du serveur) et un fichier README donnant les indications pour l'exécution. Le développement doit se faire en langage C sur une machine UNIX. Il peut être réalisé en binôme.
Vous devrez remettre une documentation décrivant le fonctionnement de votre réseau P2P, notamment :

— le protocole applicatif (type et format des messages) entre un pair et le serveur central et entre deux pairs ;
— l'algorithme général d'un pair et les structures de données utilisées ;
— l'algorithme général du serveur central et les structures de données utilisées.