Université de Rennes 1 **DESS Isa -** 1998/99

IFSIC
INSTITUT DE FORMATION SUPERIEURE EN INFORMATIQUE et COMMUNICATION

TP REDI n°1

But : apprentissage de l'interface de programmation (appelé “socket”) proposé par le système d’exploitation Unix pour réaliser des applications communicantes en utilisant les protocoles de la famille Internet.

> Latence [Petit Robert], n.f. : État de ce qui est caché, latent. Temps de latence : durée s'écoulant entre un stimulus et la réaction.

1. Présentation

On envisage de réaliser une application (répartie) permettant de mesurer le temps de latence des communications entre deux stations situées sur un réseau utilisant les protocoles de la famille Internet. L'interface de programmation de ces communications sous Unix sont des “sockets”.

Dans un premier temps, on vous demande d'utiliser les fichiers exécutables fournis puis de visiter le code proposé, ceci afin de répondre aux questions et ainsi concevoir au mieux votre application.

Dans un deuxième temps, on vous propose de réaliser cette application (en vous inspirant des fichiers mis à votre disposition) afin de répondre aux dernières questions.

Pour faciliter votre compréhension puis le développement de votre propre code, plusieurs fichiers sont mis à votre disposition : des fichiers exécutables et les fichiers de code correspondants. Ils se trouvent sous le répertoire “~dessisa/unix/redi/TP1”.

2. Apprentissage

On vous propose d'utiliser les fichiers du (sous-)répertoire “Demandeur-repondeur”. Recopiez ces fichiers. **Lisez le fichier README** : pour comprendre le fonctionnement du programme, la génération des exécutables et l'utilisation des commandes qui vous sont proposés.

Répondez aux questions suivantes à l'aide ces programmes (et de vos connaissances).

1- Quel est le temps de latence entre votre station et une des stations voisines ? Quel est le numéro de port du répondeur ? Quel est le numéro de port du demandeur ?

2- Quels sont **les** protocoles utilisés (indice : la commande “man” et le cours !) ?

3- Vérifiez qu’il est impossible à deux processus situés sur la même machine d’utiliser le même port. Quel est le message généré ? Vérifiez qu’il est possible au processus de demande de calculer le temps de latence en s’auto-envoyant une demande.

4- Pourquoi a-t-on choisi de mesurer le temps de latence sur un aller-retour et non pas simplement un seul trajet ? (indice : mise à l'heure et NTP)

5- Répétez plusieurs fois la mesure du temps de latence entre les deux mêmes stations. On constate que l'environnement à une influence sur ces mesures. Caractérisez précisément les différentes constituants de cet environnement ? Est-il facile de corriger ces inconvénients ?

6- Lorsque vous tentez de calculer pour la **toute première** fois un temps de latence vers une autre station, la durée obtenue est souvent plus importante que les fois suivantes. Pourquoi ? (Vérifiez à l'aide de la commande "arp" que c'est bien la toute première fois).

7- Les mesures sont entachées d'un certain nombre d'erreurs. Quelles sont-elles ? Proposez quelques solutions pour améliorer vos mesures (indices : place de vos points de mesure dans le code, précision de l'horloge).

8- Quand et comment devriez-vous lancer votre processus de réponse pour qu'il puisse réagir à toutes sollicitations (indice : les processus "deamon", la commande "inetd") ?

9- Que pourrait-il se passer si plusieurs processus de demande situés sur des stations différentes accèdent simultanément au même processus de réponse situé sur la même station (hypothèse : la durée de traitement du message de demande est très importante) ? Discutez de ce qui se passerait si la communication utilisait le protocole UDP (indice : la fonction fork()). Proposez une solution pour le protocole TCP (indice : les fonctions listen(), accept() et fork()).

10- Que pourrait-il se passer si plusieurs processus de demande situés sur la même station accèdent simultanément au même processus de réponse situé sur la même station ? Proposez une solution (indice : allocation dynamique). Y a t-il un problème si deux processus demandeur situés sur deux stations différentes utilise le même port ?

11- Pourquoi arrive-t-il que le processus demandeur se bloque lorsque vous lancez un très grand nombre d'échanges et que le réseau est chargé (indice : la commande "perfmeter") ? Proposez une solution.

12- Est-il possible d'utiliser le service standard d'écho pour calculer la latence ? Quel est le numéro de port associé à un tel service ? Quelle différence y a t-il entre le service rendu par "echo" et celui du répondeur ?

13- Mesurez l'influence de la longueur des messages échangés sur la latence. On vous propose de faire plusieurs fois une dizaine d'échanges pour des messages de 1, 1024, 10000, 20000, 40000 et 60000 octets. Effectuez vos mesures, entre deux stations différentes (précisez lesquelles), puis sur une seule station en auto-envoi. Tracez les courbes (en retenant les valeurs significatives), proposez une équation, calculez le débit utile.

3. Application

3.1- Présentation

On vous propose maintenant de réaliser votre propre programme. Attention : ne tenez compte que des remarques qui sont essentielles au bon fonctionnement de votre programme et qui vous ont été formulées à travers certaines des questions précédentes. Il est inutile (dans un premier temps) d'envisager un programme trop sophistiqué : vous ferez par exemple l'hypothèse qu'il n'y a pas de perte, etc. Ce programme, inspiré de celui qui vous est proposé dans le (sous-)répertoire "Demandeur-repondeur", devra utilisé **l'autre** protocole de la famille Internet.

3.2- Conception

L'application sera formée de 2 processus : un processus de demande et un processus de réponse. Le processus de demande est exécuté à chaque fois qu'un utilisateur désire mesurer le temps de latence entre une des stations du réseau munie d'un processus de réponse et la station

locale sur laquelle est exécuté le processus de demande. Le processus de réponse devra être présent, et lancé préalablement à toute tentative de calcul du temps de latence, sur chaque station susceptible de devenir une extrémité répondante de l'application.

Le processus de demande émet un message de demande vers la station munie du processus de réponse et note l'heure d'émission. Le processus de réponse lors de la réception du message de demande, émet (retourne) le message reçu vers la station ayant émis le message de demande. Le processus de demande, lors de la réception du message de réponse (s'il correspond bien à son message de demande), note l'heure d'arrivée. La différence des deux heures est, bien entendu, égale au temps de latence.

3.3- Spécification de l'interface

La commande associée au processus de réponse aura un paramètre optionnel permettant de préciser le numéro du port utilisé.

La commande associée au processus de demande aura un premier paramètre permettant de déterminer le nom de la station distante et des paramètres optionnels définissant la longueur du message échange, le nombre de messages échangés, le numéro de port du répondeur et celui du demandeur.

Le processus de demande doit afficher le nom de la station locale, le nom de la station distante suivi des temps de latence minimum, moyen et maximum en microseconde. En cas de problème, un message d'erreur explicite doit être affiché. La valeur de retour de la commande devant être négative.

3.4- Développement

On vous propose de développer les versions suivantes :

- 1.0 : version simple (une seule connexion traitée à la fois)
- 1.1 : version 1.0 avec envoi multiple de messages
- 1.2 : version 1.1 avec longueur variable des messages
- 1.3 : version 1.2 avec sous-serveurs
- 1.4 : version 1.3 avec temporisateur

Testez complètement et à chaque fois votre version avant de passer à la suivante. N'oubliez pas de sauvegarder le code avant de passer à la version suivante.

Pour assurer l'interopérabilité entre les processus des différentes équipes (un processus de demande développé par une équipe devant être compatible avec un processus de réponse d'une autre équipe), nous vous proposons d'utiliser les constantes et les formats prédéfinis dans le fichier "latence.h", puis de vérifiez cette interopérabilité.

3.5- Questions

A l'aide de votre propre programme répondez aux trois premières questions, puis à la dernière.

I- Donnez la latence minimum, moyenne et maximum pour échanger un octet entre deux stations proches. Précisez le nom de ces deux stations et les paramètres de l'expérience.

II- Donnez la latence minimum, moyenne et maximum pour échanger un octet entre deux stations plus éloignées. Précisez le nom de ces deux stations et les conditions de l'expérience.

III- Comment le répondeur sait-il que le demandeur a terminé la communication ? Vérifiez que les connexions que vous établissez sont correctement fermées. A l'aide de la commande "netstat", indiquez quels sont les différents états que prend vos connexions. A quoi correspondent-ils ? Quels sont les états qui apparaissent lorsque le répondeur n'a pas fermé son extrémité alors que le demandeur l'a fait (du coté répondeur et du côté demandeur).

IV- Quelles différences (protocoles employés, services) y a-t-il entre votre application et celle associée à la commande “ping” ?

4. Les réponses

Vous devez fournir les réponses aux question I à IV et le code de la dernière version **correcte** de votre application, sous forme électronique, dans le (sous-)répertoire “Reponses”. Consultez le fichier _README, pour plus de précisions.