TM ASR4 Téléinformatique - Programmation

A réaliser en binôme et à remettre AU PLUS TARD à 21h le lundi 12 juin 2006.

Le but de ce TM est de mettre en œuvre certaines notions décrites en cours de réseau ainsi que les techniques de programmation objets. Ainsi, le rapport à rendre est destiné à vos enseignants de ASR4-Téléinformatique et de AP2-Algorithmique et Programmation. Cependant, la programmation reste une composante essentielle de ce TM, le document devra contenir au minimum (pas forcément dans l'ordre indiqué ni pour autant dans le désordre) :

- une justification de l'implémentation des différentes classes,
- une brève description des choix retenus pour la partie Téléinformatique,
- l'état d'avancement de votre travail,
- les sources commentés en annexe.

Vous copierez également vos sources dans le répertoire `~dutour/TM-FENETRE/`*login1-login2/*. Le répertoire `~dutour/TM-FENETRE/` sera ouvert le jour de la remise du TM.

On souhaite réaliser deux versions d'un programme permettant de simuler des échanges de trames entre un émetteur et un récepteur :

Version 1. Implémentation du protocole "Envoyer et Attendre" complet (avec numérotation des trames et des acquittements, ainsi que la mise en place d'un temporisateur).

Version 2. Implémentation du protocole avec fenêtres d'anticipation utilisé dans HDLC. On suppose donc que le récepteur et l'émetteur dispose d'une fenêtre de taille maximale (en respectant les contraintes vues en TD d'ASR4).

Un programme squelette vous est fourni, vous trouverez tous les fichiers spécifiés ci-après (et un `Makefile` associé) dans `/net/exemples/ASR4/tm-fenetres`.

Les sources fournies initialement implémentent un exemple illustrant comment lancer, dans un même processus, deux objets qui s'executent simultanément. Vous n'avez pas à modifier le programme principal `main.cc` (ni à connaître la programmation multi-threads).

• On suppose donc que les deux objets `Emetteur` et `Recepteur` exécutent simultanément leur méthode `Run`. Dans cet exemple, l'objet `Emetteur` envoie une série de 26 caractères qui sont consommés par l'objet `Recepteur` 2 fois plus lentement.
La trace de l'exécution montre qu'un "timer" pour chacun des 2 objets déclenche une fonction "handler" (différente pour chaque objet) dont le traitement est réduit pour

l'instant à un affichage.

• La communication entre l'émetteur et le récepteur est "simulée" à l'aide de deux files :
- pour envoyer un message, l'émetteur fait appel à sa méthode `envoyer()` qui dépose le message dans la file `file_emetteur_vers_recepteur`, le récepteur fait appel à sa méthode `recevoir()` qui prend un message dans cette même file,
- pour envoyer un message, le récepteur fait appel à sa méthode `envoyer()` qui dépose le message dans la file `file_recepteur_vers_emetteur`, l'émetteur fait appel à sa méthode `recevoir()` qui prend un message dans cette même file.

• Pour simplifier, l'exemple permet d'envoyer un simple caractère (`char`). Mais pour ce travail, vous devez définir une nouvelle classe `Trame` permettant d'encapsuler les différentes informations nécessaires au fonctionnement du protocole :
- le type de la trame (acquittement ou donnée),
- le numéro de la trame,
- le contenu (qui peut être un simple caractère, une chaîne, ou encore mieux, le contenu binaire et son code correcteur d'erreur que vous avez implémenté dans le TM précédent).

La prise en compte de la classe `Trame` est normalement de la seule modification que vous devez apporter aux fichiers `main.cc` et `File.h`.

• Pour simuler la perte d'une trame, les méthodes `envoyer` effectuent leur traitement suivant une certaine probabilité de perte (obtenue à l'aide de la fonction `rand()`). Cette probabilité est fixée dans le constructeur des 2 objets (par défaut, il n'y a pas de perte.

• Afin de vous permettre d'implémenter le protocole demandé, un "timer" a été mis en place dans le constructeur de chacun des 2 objets. Le "timer" de l'émetteur (respectivement récepteur) déclenche (toutes les 2 secondes par défaut) la méthode `handler_emetteur` (respectivement `handler_recepteur`).

• Enfin, vous avez le choix des structures de données à utiliser pour implémenter la notion de fenêtre.

Ainsi, on vous demande d'écrire la classe `Trame`, et de compléter les classes `Emetteur` et `Recepteur` afin d'implémenter deux versions du programme : le protocole "Envoyer et Attendre" (Version 1), puis le protocole avec fenêtres d'anticipation (Version 2).