TD4 Protocoles

L'objet de ce TP est de simuler les différentes versions du protocole de liaison :

1. la version simple (*envoyer et attendre*)
2. la version sans doublon en réception (*numérotation sur 1 bit des trames, acquittement non numéroté*)
3. la version du bit alterné (*numérotation sur 1 bit des trames et des acquittements*),
4. la version avec les fenêtres coulissantes.

Seules les deux premières versions sont implémentées (fichiers `liaison-v1.cc` et `liaison-v2.cc`).

Après avoir installé la hierarchie de fichiers contenant dans `~pecher/TI/nachos-reseau.tar.gz`, placez-vous dans le répertoire `nachos_reseau/code/reseau`.

Lancer la commande `make`. Lancer un simulateur du récepteur `./nachos -m 1 -l 1` et lancer *rapidement* (sous 3 secondes), dans une autre fenêtre, le simulateur de l'émetteur `./nachos -m 0 -l 1`.

Exercice 1 : prise en main de `nachos`

Observer le comportement du simulateur : observe-t-on des doublons, des pertes de messages ?

Utiliser le mode debug `-d l` pour tracer plus finement l'évolution du protocole.

Le paramètre `-l` permet de faire varier le taux de perte des messages émis par le simulateur : `-l 1` signifie une liaison parfaite et `-l 0.9` signifie qu'une émission sur dix sera perdue en moyenne. Faites les expériences suivantes :

Récepteur	Emetteur
`./nachos -m 1 -l 1 -d l`	`./nachos -m 0 -l 0.7 -d l`
`./nachos -m 1 -l 0.7 -d l`	`./nachos -m 0 -l 1 -d l`
`./nachos -m 1 -l 0.7 -d l`	`./nachos -m 0 -l 0.7 -d l`

Il est aussi possible de faire varier la durée de temporisation en multipliant ou en divisant la valeur `TEMPO_ACQUITTEMENT` de la procédure `CoucheLiaison::Emettre`. Quels sont les effets d'une temporisation plus longue ? Et ceux d'une temporisation courte ?

Observer le fichier `liaison-v1.cc`. Une procédure particulière (`DemonReception`) est exécutée par un thread sur chaque machine. Son travail consiste à transformer en événement toute arrivée de trame et de placer systématiquement celle-ci dans un buffer. Vous n'aurez pas à la modifier.

Le cœur de la couche liaison est constitué de deux fonctions (elles aussi exécutées par des threads) : `ProtocoleEmission` et `ProtocoleReception`. Leur rôle est de réagir aux événements (arrivée d'une trame à émettre, réception d'une trame ou d'un acquittement, alarme déclenchée par un temporisateur) conformément au protocole en vigueur. Votre travail d'aujourd'hui consistera essentiellement à modifier ces deux fonctions pour étendre les services rendus par la couche liaison.

Exercice 2 : version sans doublon

Donner sur le papier une esquisse du code à ajouter pour mettre en place la numérotation alternée des trames émises (côté émetteur).

Valider ensuite votre esquisse en analysant le fichier `liaison-v2.cc`.

Exercice 3 : test

Entrer `make V=2` puis reprendre les exemples du premier exercice afin d'analyser le comportement du protocole sans doublon.

Exercice 4 : bit alterné

Mettre en œuvre le protocole du bit alterné en numérotant aussi sur un bit les trames d'acquittement. À cette fin, on créera le fichier `liaison-v3.cc` que l'on compilera via la commande `make V=3`.
Vérifiez le bon comportement de votre protocole.

Exercice 5 : fenêtre coulissante

Mettre en œuvre le protocole à fenêtre coulissante sans rejet sélectif : il s'agit de pouvoir envoyer un nombre prédéterminé de trames sans avoir nécessairement reçu l'acquittement de la première. On utilisera en émission un tampon des trames émises et non encore acquittées.