Magistère STIC — 2006–2007
Module *Réseaux* — Feuille de TP n° 2

Protocoles de liaison de données

Dans ce TP nous allons réaliser certains des protocoles de liaison de données discutés en cours.

Copiez les fichiers qui se trouvent dans le répertoire `~rtreinen/reseaux/tp2`. Vous y trouverez

- `threadtop` : un interprète OCaml avec les fonctionnalités nécessaires pour exécuter des programmes avec des threads.
- `simulator.ml` : le code OCaml pour la simulation des couches physiques et de réseau.
- `multitime.ml` : le code OCaml pour gérer plusieurs timers concurrents.
- `protocole1.ml` : le code OCaml pour le premier protocole (pour un canal idéal et sans contrôle de flux)

Exercice 1 (Un protocole avec contrôle de flux)
Coder un protocole de communication avec un contrôle de flux : l'émetteur attend l'arrivée d'un acquittement avant d'envoyer une nouvelle trame. Dans cette exercice vous pouvez toujours supposer un canal idéal sans perte et sans erreur.

Exercice 2 (Un protocole pour un canal bruité)
Coder un protocole de communication avec contrôle de flux et pour un canal bruité, en utilisant un timer et un compteur pour numéroter les trames. Utiliser la fonction `create_multitimer` du fichier `multitimer.ml` pour créer un (ou plusieurs) timers.

Exercice 3 (Un protocole avec fenêtre d'anticipation)
Coder le protocole de communication avec fenêtre d'anticipation. Vous pouvez supposer que l'émetteur et le récepteur utilisent la même taille pour leur fenêtre. Dans un premier temps réaliser un protocole dans lequel le récepteur n'envoie que des acquittement positives.

Utiliser dans cet exercice la fonction `create_network_agent` du fichier `simulator.ml`. Cette fonction crée un thread pour la couche réseau de l'émetteur. La fonction prend comme arguments le délai entre l'envoi de deux paquets, un un signal qui est utilisé pour indiquer la disponibilité d'un nouveau paquet. Elle envoie des fonctions pour arrêter le thread et pour le relancer, et une fonction pour récupérer un paquet.