Magistère STIC | 2006–2007
Module *Réseaux* | Feuille de Examen

Examen du cours *Réseaux*

Durée : 3 heures
Documents autorisés : Notes de cours et notes personnelles, copies des transparents du cours, énoncés et solutions des TP. Les livres et les ordinateurs ne sont pas autorisés.
Le barème est indicatif.

Exercice 1 (2 points)
Dans un article récent du *Monde* on pouvait lire que « tout ordinateur possédant une connexion réseau peut être identifié de façon unique par son numéro IP » (cité de mémoire). On peut lire cette phrase de deux façons :
- « un ordinateur utilise toujours le même numéro IP pour se connecter »
- « il n'est pas possible que deux ordinateurs différents utilisent le même numéro IP »

La phrase citée est fausse dans les deux interprétations. Expliquer pourquoi (il y a plusieurs raisons).

Exercice 2 (3 points)
On considère un réseau consistant de routeurs et de machines utilisateurs. Les routeurs calculent leurs tables de routage par « routage par informations d'états de lien » (link state routing). Supposons que le coût d'un lien est exprimé par un entier positif, et que tout lien peut être traversé avec le même coût dans les deux sens.

Donner un exemple d'un tel réseau qui contient au moins deux machines utilisateur *Alice* et *Bob* ; et plusieurs routeurs dont un (nommé *Charlie*) est malhonnête, tel que
- le chemin le plus court entre *Alice* et *Bob* ne passe pas par *Charlie*,
- mais *Charlie* peut envoyer des informations de lien faussées telles que les messages envoyés de *Alice* vers *Bob* passent par lui même.

Détaillez sur votre exemple quelles sont les paquets d'informations envoyés par *Charlie.*

Exercice 3 (3 points)
Supposer que vous est censé concevoir le format d'entête d'un paquet de la couche réseau. Vous utilisez CRC pour calculer le total de contrôle (checksum) d'une entête. Parmi les champs de l'entête il y en a un qui change à chaque passage par un routeur. Chaque routeur va donc être obligé de mettre à jour le total de contrôle.

Quel est le meilleur endroit pour ce champs dans l'entête qui permet une mise à jour efficace du total de contrôle ? Expliquer pour votre solution comment le routeur peut mettre à jour le total de contrôle.

Exercice 4 (2 points)
Parmi les méthodes d'authentification utilisées par l'outil `ssh` (secure shell) il y a l'authentification par paire de clefs. La clef privée est stockée sur le disque local du client et est verrouillée ; un utilisateur doit entrer un mot de passe pour déverrouiller la clef privée et de l'utiliser pour l'authentification.

Un utilisateur va donc être obligé d'entrer un mot de passe. Pourquoi est-ce que cette méthode d'authentification est quand-même plus sûre que l'authentification par mot de passe comme réalisé par des protocoles comme `telnet` ou `rlogin` ?

Exercice 5 (10 points)
Programmez un protocole de communication avec fenêtre d'anticipation dans le style du TP numéro 2. Le programme doit réaliser une communication sans perte et sans duplication de trames sur un canal qui peut perdre des trames. La taille maximale de la fenêtre de l'émetteur est donnée par une constante n_e, et celle du récepteur par m_r. Votre récepteur a le droit d'envoyer un seul message d'acquittement après réception d'une trame. Réalisez un algorithme dans lequel le récepteur acquitte toujours le numéro de trame t le plus haut tel qu'il a bien reçu toutes les trames jusqu'au numéro t inclus.

Votre programme consistera d'une partie exécutée par l'émetteur, et d'une partie exécutée par le récepteur. Écrire votre programme en OCaml (les détails de la syntaxe OCaml, et les noms exactes des fonctions de bibliothèque n'importent pas ici).

Nous pouvons ici ignorer certaines problèmes qui dans le TP étaient dûs au fonctionnement du simulateur. Ainsi, utiliser

- la fonction `from_network_layer: unit -> string` pour obtenir un paquet de la couche réseau,
- la fonction `to_network_layer: string -> unit` pour délivrer un paquet à la couche réseau,
- la fonction `send_frame: frame -> unit` pour envoyer une trame sur le canal. C'est à vous de définir le type `frame`.
- la fonction `get_frame: unit -> frame` pour récupérer une trame disponible à la sortie du canal. Déclenche une exception `No_frame` quand aucune trame est disponible.
- les fonctions `send_ack: ack -> unit` et `get_ack: unit -> ack` qui sont analogues pour le canal des acquittements. C'est à vous de définir le type `ack` des acquittements.
- la fonction `start_timer: int -> unit` pour démarrer une horloge d'alarme (timer) avec un certain numéro, la fonction `stop_timer: int -> unit` pour l'arrêter—. Les horloges qui déclenchent leur alarme sont mises dans une file d'attente ; la fonction `get_alarm: unit -> int` renvoie l'entrée la plus vieille de cette file et supprime cette entrée de la file. Cette fonction lève une exception `No_alarm` quand la file est vide.

Ne vous souciez pas de la gestion des verrous (Mutex).