# 1 Pour aller plus loin .... des exercices d'examens

## Exercice 1

Une méthode pour empêcher un message de vivre trop longtemps dans les réseaux est d'utiliser la limite du nombre de routeurs que ce message peut traverser (champ *TTL* dans un paquet IP - voir cours). Une valeur par défaut est fixée au lancement du système, mais elle peut être modifiée par du logiciel du niveau de la couche réseau. Ainsi, au départ de tout paquet la durée de vie est fixée et permet en particulier d'éviter les boucles de routage.
Cette caractéristique est utilisée conjointement avec le paquet d'erreur qui est retourné, pour déterminer l'existence d'un chemin pour atteindre une destination (cf. `traceroute`).

1. Rappeler très brièvement comment fonctionne un tel logiciel de détermination de chemin.
2. Pourquoi est-ce que le paquet d'erreur ne peut être retourné qu'à la source ?
3. Pourquoi est-ce que le chemin ainsi trouvé peut s'avérer
   (a) faux,
   (b) inexistant ?
4. En supposant qu'aucune nouvelle panne ne se produit pendant un intervalle de temps, est-il possible, sans modifier le protocole IP, d'obtenir un chemin certain ?

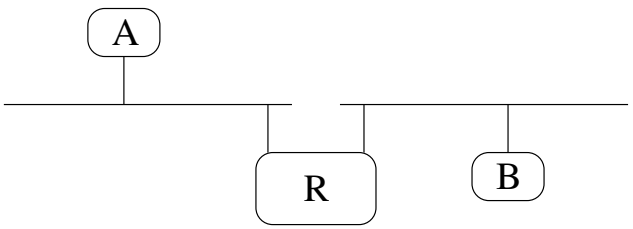
## Exercice 2

On considère ici une application de transfert de documents. Pour fixer les idées, on peut prendre l'exemple de *W3* avec le protocole de transfert de documents hypertexte, *Http*.
On suppose d'abord qu'un document est représenté par une suite de caractères.
Pour recevoir un document, une application de type *Client* demande l'établissement d'un circuit virtuel avec un *Serveur* par le biais du protocole *tcp*. Une fois la connexion établie, le client envoie la référence du document (selon le protocole d'application convenu) et le serveur lui renvoie ce document. On suppose enfin que le serveur ferme la connexion à la fin de l'expédition. Le client se contente de lire la suite de caractères jusqu'à la réception de l'information de fermeture.

1. Est-ce que le serveur peut expédier le document en une seule instruction d'expédition ? est-ce que le client peut recevoir tout le document en une seule lecture ? est-ce que le client peut déduire le nombre de paquets *tcp* qui ont été acheminés ?
2. Rappeler très rapidement comment est reçue l'information de fermeture chez le client, à savoir, sous l'aspect programmation, comment est-ce que le client récupère cette information ; ajouter une explication brève.
3. Montrer que malgré la fiabilité de *tcp* le client n'est pas sûr d'avoir reçu la totalité du document.
4. Proposer une solution pour qu'il puisse le savoir.
5. Ajouter une solution permettant de récupérer par la suite la partie manquante seule.
6. On suppose maintenant qu'un document est composé de plusieurs parties différentes. Chaque partie peut être vue comme un document en soi (une image par exemple). Proposer une solution permettant une reprise «au mieux» en cas de réception partielle.

## Exercice 3

Dans la configuration ci-après, concernant deux segments *ethernet*,




voici comment ont été configurés les deux hôtes A et B et le routeur R :

**hôte A :**

| | |
|---|---|
| **adresse IP :** | 180.220.129.129 |
| **masque :** | 255.255.0.0 |
| **routeur par défaut :** | 180.220.129.1 |

**hôte B :**

| | |
|---|---|
| **adresse IP :** | 180.220.193.193 |
| **masque :** | 255.255.0.0 |
| **routeur par défaut :** | 180.220.193.1 |

**routeur R :** deux adresses IP : 180.220.129.1 sur le brin gauche de la figure et 180.220.193.1 sur le brin droit.

1. Quel défaut peut-on remarquer dans cette configuration ? Pour répondre, envisager un paquet IP partant de A à destination de B. Comme l'évolution d'un paquet a été vue ci-avant, il n'est pas nécessaire de le développer ici. Il faut indiquer uniquement les éléments engendrant le défaut.
2. Quelle(s) correction(s) faut-il effectuer ? Ne proposer qu'une seule solution cohérente. Développer après correction l'évolution du paquet ci-dessus.

## Exercice 4

On veut empêcher un même client de faire plusieurs demandes de connexion successives à un même serveur. Par exemple, éviter qu'un moteur de recherche surcharge un serveur par des requêtes en rafale, empêchant toute autre demande de connexion à ce serveur.

1. Quelles sont les informations permettant d'identifier un client ? Analyser dans votre réponse toutes les configurations d'un serveur, quel que soit son type, concurrent ou itératif, utilisant `UDP` ou `TCP`.
2. Proposer une solution permettant au serveur de refuser plus de $n$ demandes de connexion par minute. On pourra admettre l'existence d'une fonction de mesure du temps (choisissez une mesure qui vous convient, admettez son existence). Décrire ce que fait le serveur et ce qui se passe chez le client.
3. Est-ce que votre solution empêche le client de saturer la file d'attente dans le cas d'un serveur en mode connecté ? dans le cas d'un serveur en mode sans connexion ?