1 Deuxième partie

1.1 Quels exemples peut-on proposer ? On peut proposer ici des exemples qui pourront facilement être mis en œuvre en TP, et d'autres cas possibles, qu'on ne pourra pas mettre en évidence.

L'objectif, ici, n'est pas seulement de lister l'ensemble des pannes possibles, mais également d'indiquer si ces dernières sont facilement implémentables dans le cadre du TP. Voici une liste non exhaustive :

— Pannes implémentables :

1. Saturation du buffer de la socket du serveur. En effet, la taille mémoire d'un buffer étant de l'ordre de 8ko, il suffit de mettre le serveur en attente assez longtemps, pendant qu'un (ou plusieurs) client envoient des messages en boucle.
2. Fermeture de la socket en réception côté serveur. De la même manière qu'un pipe, une socket est bidirectionnelle (elle peut servir en réception et en émission), et comme le pipe, il est possible de ne fermer qu'un côté de la socket, sans la fermer totalement.

— Pannes non implémentables :

1. Pannes physiques : tout ce qui touche à une panne matérielle (ou même à sa simulation) n'est pas autorisé par le règlement intérieur (et la charte informatique) de l'université.
2. Saturation du réseau : on peut imaginer de saturer suffisamment le réseau de requête que le routeur (pour diverses raisons) sera susceptible de *tomber*, c'est le principe de l'attaque par déni de service (qui au passage est légalement répréhensible).
3. Problème de routage : une mauvaise table de routage sur un routeur est susceptible de mener à une perte du paquet, naturellement mettre en place une telle panne nécessite les droits d'administration sur les routeurs de la fac...

1.2 Quelles sont les modifications à apporter aux programmes afin de mettre en évidence une telle perte ?

Ici, le but est de proposer un schéma algorithmique permettant de simuler une des pannes présentées ci dessus. La plus simple a mettre en place est la saturation du buffer, c'est donc celle-ci que nous choisirons.

1.2.1 Côté serveur

Comme évoqué précédemment, il faut mettre le serveur *au repos* le temps pour le client de saturer sa socket.

Algorithme 1 : Schéma algorithmique du serveur

```
Allocation socket_locale;
Initialisation socket_distante;
Sleep 30s;
while True do
    Lire Message sur socket_locale;
    Afficher Message;
```

1.2.2 Côté client

Ici aussi pas de grands changements, juste besoin d'envoyer en boucle vers le serveur sans attendre de réponses.

Algorithme 2 : Schéma algorithmique du client

```
Allocation socket_locale;
Initialisation socket_distante;
while temps < 30s do
    Envoyer message sur socket_distante;
```

1.3 Est-il possible de créer un exemple d'interblocage des deux applications (au besoin en modifiant les programmes) ?

Comme son nom l'indique, pour qu'il y ait interblocage, il faut que le client soit en attente d'une action du serveur et que le serveur soit en même temps en attente d'une action du client. Toute solution consistant à faire en sorte que le client n'envoie pas de messages mène à un blocage du serveur mais en aucun cas à un interblocage, puisque le client n'est pas bloqué.

Une solution simple peut être de mettre le serveur et le client en attente d'un message en même temps, c'est facilement faisable en utilisant un seul et même code pour le client et le serveur :

Algorithme 3 : Schéma algorithmique du client et du serveur illustrant un interblocage

```
Allocation socket_locale;
Initialisation socket_distante;
Lire message sur socket_locale;
Envoyer message sur socket_distante;
```

1.4 Est-il possible alors de les débloquer par des moyens autres que l'arrêt pur et simple ?

Oui, naturellement. Il suffit d'utiliser un programme tiers pour envoyer un message à l'un (voire aux deux) des protagonistes sur le port de la socket en écoute.

2 Troisième étape

2.1 Que proposez vous pour la mise en place d'un dialogue sans heurts ?

Il est inutile de réinventer la roue, la fonction `recvfrom` prend en paramètre un pointeur sur `struct sockaddr` qui sera rempli avec les informations de la socket ayant envoyé le message. Du coup, il suffit simplement d'utiliser cette structure dans le `sendto` en guise de destinataire.