

Université de Bretagne Occidentale  
Licence Informatique option informatique

## Examen de TP de réseaux

Mardi 20 mai 2003, 13h30-15h30

---

### Documents autorisés

Notes de cours, de travaux dirigés et de travaux pratiques. Accès aux comptes personnels interdit.

### Remarques

Lire le sujet en entier avant de commencer l'examen. La rédaction de commentaires dans les codes des programmes sera appréciée par le correcteur. Le barème est donné à titre indicatif.

## 1 Contexte de l'exercice

Le protocole de communication UDP est un protocole non fiable, c'est à dire que l'émetteur d'un message n'a aucune garantie quant à la réception de son message.

Dans le cadre d'une application utilisant ce protocole, un développeur souhaite, de temps en temps, vérifier que le destinataire de ses messages les reçoit effectivement. Il envoie donc des messages spécifiques qui demandent la production d'un acquittement par le destinataire. Si au bout d'un temps déterminé, aucun acquittement n'est fourni, l'émetteur réessayera un nombre de fois fini. Si dans cette phase, aucun acquittement n'est reçu, l'émetteur considérera alors que son destinataire n'est plus joignable et cessera donc ses émissions de message.

## 2 Réalisations demandées

### 2.1 Fonctionnement sans traitement de l'acquittement (8 pts)

Construire :

- Un programme émetteur, qui envoie une série de 10 messages contenant la chaîne "DATA", puis qui envoie un message contenant la chaîne "DATA\_ACK" puis qui attend une réponse de son destinataire (cette attente peut-être infinie, en cas de perte du message ou de son acquittement).

On supposera que le programme émetteur répète cette séquence éternellement.

- Un programme récepteur, qui reçoit éternellement les messages en provenance de l'émetteur et qui ne répond qu'aux messages contenant la chaîne "DATA\_ACK" par un message contenant la chaîne "DATA\_RECEIVED".

Pour cette partie, les programmes devront impérativement être nommés : `emetteur1.c` et `recepteur1.c`. Des explications pourront être fournies dans un fichier de type texte nommé : `comment.txt`

## 2.2 Fonctionnement avec traitement de l'acquittement (6 pts)

Construire :

- Un programme émetteur, qui envoie une série de 10 messages contenant la chaîne "DATA", puis qui envoie un message contenant la chaîne "DATA\_ACK" puis qui attend une réponse de son destinataire. Si au bout de cinq secondes, aucun acquittement n'est parvenu, le programme réémettra la chaîne "DATA\_ACK" une seconde fois. Si au bout de cinq secondes, aucun acquittement n'est parvenu, le programme réémettra la chaîne "DATA\_ACK" une troisième fois. Si au bout de cinq secondes, aucun acquittement n'est parvenu, le programme se terminera.

On supposera que le programme émetteur répète cette séquence éternellement tant que le cas exposé ci-dessus ne se produit pas.

Remarque : dans beaucoup d'algorithmes utilisant des chiens de garde (timeout), la durée d'attente n'est pas constante, mais multipliée par 2 à chaque itération.

- Un programme récepteur, qui reçoit éternellement les messages en provenance de l'émetteur et qui ne répond qu'aux messages contenant la chaîne "DATA\_ACK" par un message contenant la chaîne "DATA\_RECEIVED". Afin de pouvoir simuler une perte de message, l'envoi de la chaîne "DATA\_RECEIVED" devra être préalablement confirmé ou non par l'utilisateur.

Pour cette partie, les programmes devront impérativement être nommés : `emetteur2.c` et `recepteur2.c`. Des explications pourront être fournies dans un fichier de type texte nommé : `comment.txt`

## 2.3 Discussion (6 pts)

Le fonctionnement décrit dans la partie 2.2 peut entraîner des dysfonctionnements, si la réémission d'un message "DATA\_ACK" se déroule en même temps que l'émission du message "DATA\_RECEIVED" correspondant au "DATA\_ACK" précédent. Ce problème est facilement simulable avec les programmes précédemment construits, en renvoyant le message "DATA\_RECEIVED" plus de 5 secondes après sa réception.

Expliquez pourquoi et proposez une ou plusieurs solutions permettant de résoudre ce problème.

Implémentez ensuite l'une des solutions.

La réponse à cette question devra être rédigée dans un fichier de type texte nommé : `comment.txt`. Les programmes devront être nommés : `emetteur3.c` et `recepteur3.c`.

## 3 Squelette de programme

Vous trouverez sous `/home/enseignants/leparc/Licence/Reseaux/TP3`, des squelettes de programme correspondant au TP sur UDP.

## 4 Utilisation d'horloge - rappel cours systèmes d'exploitation

```
...
#include <signal.h>
...

/* handler du timer */
void timerDepasse ()
{
/* ne rien faire, le déclenchement du timer va automatiquement provoquer une
/* interruption sur les lectures bloquantes et provoquer la mise de la
/* variable errno à EINTR */
return ();
}

...

/* Programme principal */

main(int argc, char **argv)
{
/* declarations */

struct sigaction timeout;
...

/* mise en place du handler du timer */
timeout.sa_handler = &timerDepasse;
timeout.sa_flags = 0;
sigemptyset(&timeout.sa_mask);
if (sigaction (SIGALRM, &timeout, NULL) != 0)
{
perror("sigaction");
exit(-1);
}
...

/* mise en route du timer pour 5 secondes */
alarm(5);

...

/* arrêt du timer si un message est reçu avant les 5 secondes */
alarm(0);
```