Antoine Foucault
Réseau
11/12/2013

Rapport des TP n°4-5

Sommaire

1. Introduction :

L'objectif de ces deux TP sera de réaliser une application Client-Serveur.

La partie cliente sera chargée de transmettre au serveur des données reçues via deux capteurs et la partie serveur sera chargée de sauvegarder ces données tout en gérant plusieurs clients.

2. Messages :

Afin de communiquer, les applications utiliseront les structures de messages décrites ci-dessous :

REQUETE
Température (float)
Pression (float)
Numéro d'envoi (int)

REPONSE
Code de retour (int)
Numéro d'envoi retourné (int)

L'usage du numéro d'envoi est justifier par le fait que l'application cliente attend un accusé de réception lors de l'envoi d'une requête. Le numéro d'envoi permet donc de vérifier à quel requête est associé un numéro d'envoi et donc d'agir en conséquence. De plus, dans notre implémentation, le champs « numéro d'envoi » sert à notifier le serveur de la déconnexion du client s'il est placé à -1.

Le code retour n'est pas nécessaire pour satisfaire les demandes du sujet, cependant il est implémenté dans l'optique où l'application client peut vouloir savoir si la réception des données à été effectuée correctement par le serveur.

3. Partie Cliente :

La partie cliente est chargée de récupérer des informations envoyées par deux capteurs à intervalle de 20 secondes et de les envoyer au serveur. Un système d'accusé de réception est implémenté afin de rendre plus fiable la transmission des données.

3.1. Organisation de l'application :

L'illustration n°1 décrit le fonctionnement de la partie cliente.

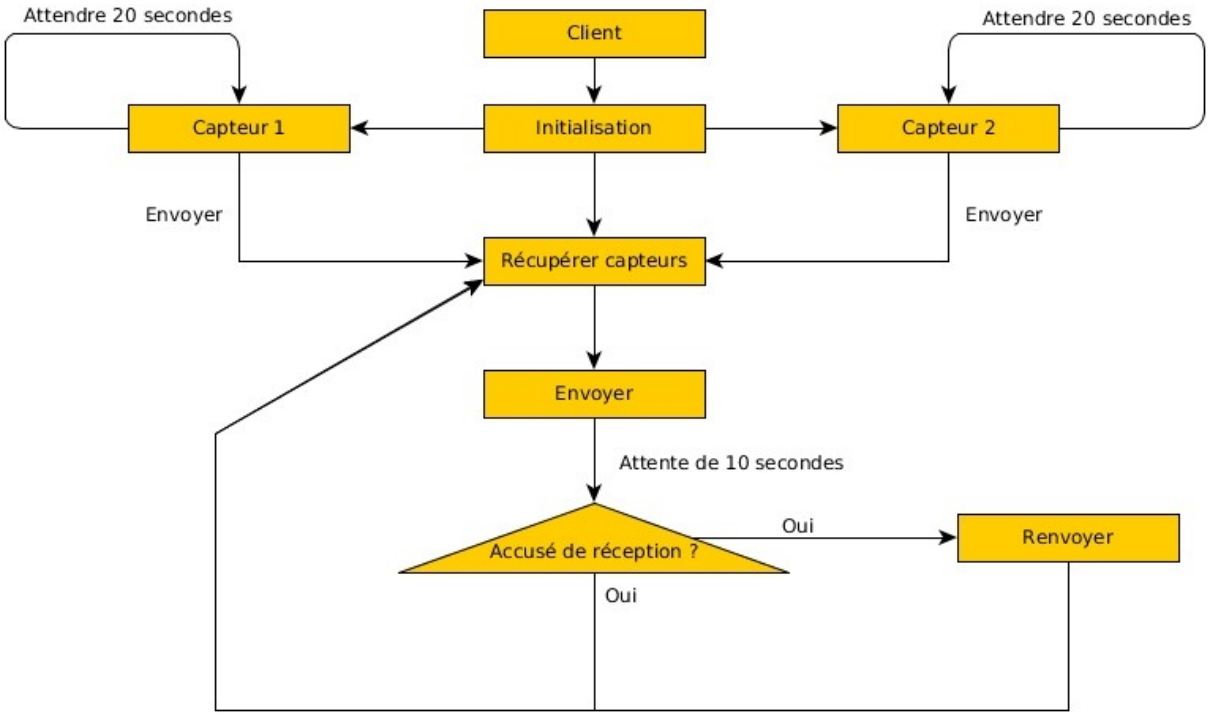

Illustration 1: Fonctionnement de l'application cliente, ce diagramme ne comprend pas le cas de fermeture du programme par l'utilisateur qui envoi un message de déconnexion au serveur

4. Partie Serveur :

La partie serveur est chargé de réceptionner les données envoyée par le client et de les sauvegarder dans des fichiers. L'application est capable de gérer plusieurs clients simultanément.

4.1. Organisation de l'application :

L'illustration n°2 décrit le fonctionnement de la partie serveur.

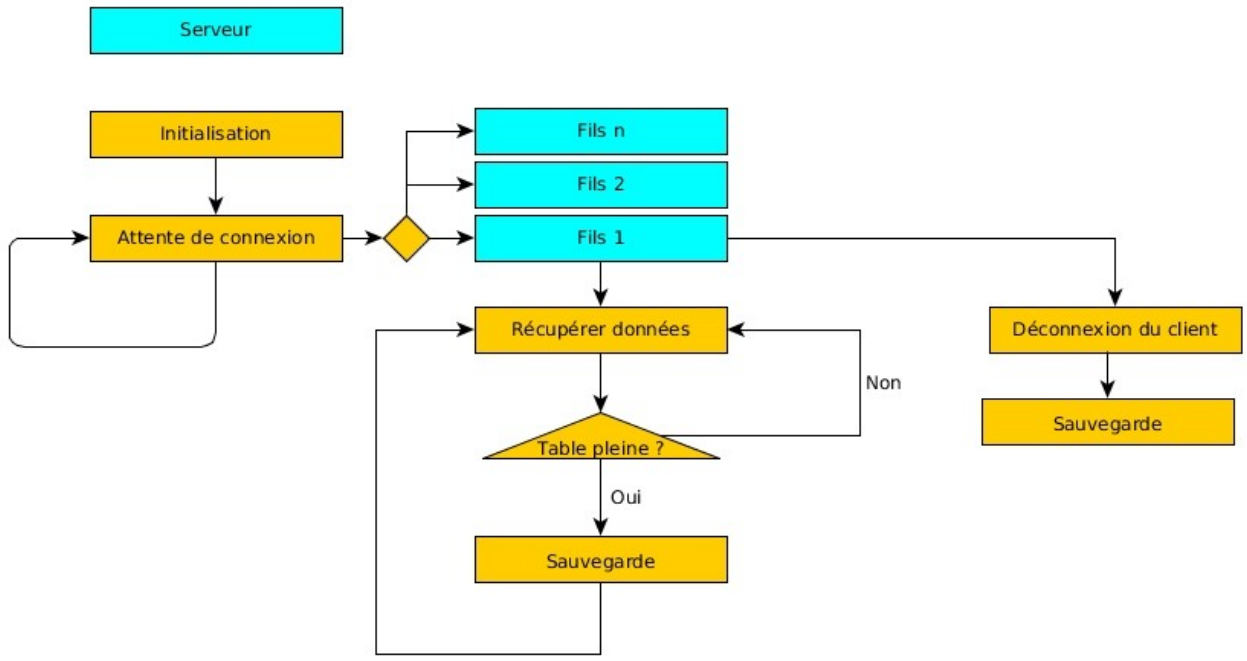

Illustration 2: Fonctionnement de la partie serveur

5. Tests :

Afin de vérifier le fonctionnement des deux parties plusieurs tests ont été effectués :

- Transfert par le client d'un nombre de données supérieur à la table d'entrée du serveur afin de vérifier la sauvegarde
- Arrêt « normal » du client pendant une connexion avec le serveur
- Arrêt anormal du client (crash, déconnexion...) pendant une connexion avec le serveur
- Envoi de données erronées au serveur (vérification du code retour)
- Non génération d'un accusé de réception par le serveur afin de tester le renvoi des données par l'application cliente
- Vérification des fichiers de sauvegarde par analyse avec un éditeur hexadécimal (hexedit)
- Vérification des communications avec Wireshark

L'ensemble des test a été jugé satisfaisant dans la réaction des applications ou l'analyse.

L'illustration 3 représente un échange d'informations entre le client et le serveur capturé à l'aide de WireShark et du filtre de capture présenté ci-dessous. Le contenu de cette capture est disponible dans le dossier « Annexes » fournit avec ce rapport.

192.168.61.151	192.168.61.12	TCP	74 39671 > 10500 [SYN] Seq=0
192.168.61.12	192.168.61.151	TCP	74 10500 > 39671 [SYN, ACK]
192.168.61.151	192.168.61.12	TCP	66 39671 > 10500 [ACK] Seq=1
192.168.61.151	192.168.61.12	TCP	78 39671 > 10500 [PSH, ACK]
192.168.61.12	192.168.61.151	TCP	66 10500 > 39671 [ACK] Seq=1
192.168.61.12	192.168.61.151	TCP	74 10500 > 39671 [PSH, ACK]
192.168.61.151	192.168.61.12	TCP	66 39671 > 10500 [ACK] Seq=1
192.168.61.151	192.168.61.12	TCP	78 39671 > 10500 [PSH, ACK]
192.168.61.151	192.168.61.12	TCP	66 39671 > 10500 [FIN, ACK]
192.168.61.12	192.168.61.151	TCP	66 10500 > 39671 [ACK] Seq=9

Illustration 3: Échanges entre l'application cliente et serveur

On remarque dans l'illustration 3 en rouge l'établissement de la connexion entre les deux application.

On constate ensuite en vert l'échange des données entre le client et le serveur :

- Une requête d'envoi des données des capteurs
- Une réponse du serveur
- Une nouvelle requête du client (première ligne en bleu)

Enfin on peut voir en bleu (deux dernières lignes) la phase de déconnexion entre les deux applications.

L'illustration 4 présente le contenu hexadécimal de l'échange entre les deux applications :

```
00000000  00 00 00 00 00 00 24 42  00 80 3e 43        ......$B ..>C
    00000000  c8 00 00 00 00 00 00 00                  ........
0000000C  ff ff ff ff 00 00 24 42  00 80 3e 43        ......$B ..>C
```

Illustration 4: Contenu de l'échange entre le client et le serveur

On peut voir en rouge le champs correspondant au numéro d'envoi, d'abord positionné à 0 par le client puis positionné à -1 pour la déconnexion (0xFFFFFFFF).

En vert on peut observer le code retour du serveur qui est dans notre exemple de 200 (0xc8).

Enfin en jaune on voit les données des capteurs qui sont stockées sur la taille de deux floats au total (4*2 octets)

6. Conclusion :

Ce TP nous a montré qu'il est important de définir les fonctionnements et les échanges entre deux applications de façon complète et claire afin d'assurer un développement serein. De plus les notions de programmation système et programmation TCP ont été revues lors du développement.

Plusieurs améliorations seraient possibles concernant cette application :

- Intégrer un système de renvoi du message en cas de non réception de l'accusé de réception sans bloquer l'envoi des nouveaux messages (application cliente plus fiable dans la transmission des données)
- Intégrer un système d'analyse des erreurs côté capteur en cas de code retour signifiant une erreur
- Créer une application chargée de traiter en local les fichiers sauvegardés par le serveur (affichage par exemple)