Hassan Paziaud SN2

Théo Bilhaut

Jean Leduc

**Travaux pratiques : Client / Serveur TCP/IP avec Qt**

**Sommaire**

[**Introduction :** 2](#_Toc148375251)

[**Questions Préliminaires :** 2](#_Toc148375252)

[**Principes de fonctionnement des applications :** 3](#_Toc148375253)

[**Rôles membres :** 4](#_Toc148375254)

# **Introduction :**

Ce TP vise à plonger dans le développement d'une application client-serveur en langage C++. L'objectif de ce TP est de concevoir deux applications. La première, une application cliente et une deuxième, une application serveur. Ces applications seront responsables de la communication à travers un réseau.

## **Questions Préliminaires :**

**Principe de la notion de client/serveur en informatique ?**

En informatique, le modèle client/serveur est une structure où les ordinateurs communiquent. Il y a 2 rôles : le client et le serveur. Le client c’est celui qui envoie les informations tandis que le serveur est celui qui écoute/reçoit les informations.

**Qu'est-ce qu'un protocole ? À quoi sert-il ?**

Un protocole sert à définir comment les données doivent être échangées et traitées entre les différentes entités. Ils sont là pour assurer une communication cohérente et fiable

**Expliquer la notion de port et de socket sous TCP/IP :**

Un port est un numéro d'identification. Les ports permettent de diriger les données vers des services spécifiques.

Un socket permet qu’un programme envoie et reçoit des données via un réseau, en utilisant des protocoles TCP/IP.

**En utilisant l'aide de Qt, quelles sont les classes de Qt permettant la création d'une application cliente et d'une application serveur ?**

Pour une application cliente c’est `QTcpSocket` et pour une application serveur, c’est `QTcpServer`.

### **Principes de fonctionnement des applications :**

**Client :**

Ce code est une application client, le client peut se connecter à un serveur et envoyer des messages. Voici un aperçu de son fonctionnement :

Connexion : la fonction onConnectButtonClicked

Gestion de la connexion : Les fonctions onSocketConnected et onSocketDisconnected.

Envoi de messages : Lorsqu'un utilisateur envoie un message, la fonction onSendButtonClicked est appelée.

Réception de messages : Lorsqu'un message est reçu, la fonction onSocketReadyRead est appelée.

**Serveur :**

La partie serveur de notre programme est une application qui écoute sur un port spécifique (dans cet exemple, le port 12345) et répond aux demandes des clients qui se connectent à ce port.

Classe du serveur Server :

La classe Server est une sous-classe de QTcpServer qui gère les connexions entrantes des clients.

Méthode incomingConnection :

Cette méthode est appelée chaque fois qu'un nouveau client se connecte au serveur. Elle crée un nouveau socket pour gérer cette connexion entrante, établit les connexions de signaux pour ce socket (pour la lecture et la déconnexion), et l'ajoute à la liste des sockets gérés par le serveur.

Méthode readyRead :

Cette méthode est appelée chaque fois qu'un client envoie des données au serveur. Elle lit les données envoyées par le client et les stocke dans requestData.

Traitement des demandes du client :

Le code commence par vérifier si la demande du client correspond à l'un des trois formats attendus : Tdxx?, Tfxx?, Hrxx?. En fonction du format, il génère une réponse en conséquence. Par exemple, s'il s'agit d'une demande de température en degrés Celsius (Tdxx?), le serveur génère une température aléatoire entre -20.00 et +35.00 et crée une réponse au format Tdxx,YXX.XX. La réponse est ensuite envoyée au client via le socket du client.

Journalisation :

Le code utilise qDebug pour afficher la demande du client et la réponse générée dans la console de journalisation du serveur. Cela permet de suivre et de déboguer les opérations.

Méthode disconnected :

Cette méthode est appelée lorsque le client se déconnecte du serveur. Elle supprime le socket du client de la liste des sockets gérés par le serveur.

Boucle principale :

Le serveur utilise une boucle principale pour rester actif et écouter les connexions entrantes sur le port spécifié (12345 dans cet exemple). Ainsi, le serveur attend des demandes de clients, les traite en fonction du protocole défini (température en degrés Celsius, température en degrés Fahrenheit, ou hygrométrie), génère une réponse appropriée, envoie la réponse au client et affiche des informations de journalisation pour chaque demande. Cela permet au serveur de fournir des réponses aux clients en fonction de leurs demandes et de suivre les interactions dans la console de journalisation.

#### **Rôles des membres :**

Hassan Paziaud : Membre effectuant le code serveur.

Théo Bilhaut : Membre effectuant le code client.

Jean Leduc : Membre aidant les 2 autres membres en faisant en plus le compte rendu.

##### **Lien Github :**

<https://github.com/tbilhaut/TD-tcp-server>

###### **Conclusion :**

Pour conclure, ce TP nous a permis d'en apprendre sur le domaine de la programmation réseau en utilisant le langage C++ avec Qt. Au cours de ce projet, nous avons conçu une application client et une application serveur.