MODULE 06
SÉANCE SYSTÈME 05
TP D'INFORMATIQUE
Durée 2h30

Le TCP RS sur Raspberry

BLOC DE COMPÉTENCES
U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)
C08 - CODER

OBJECTIF PÉDAGOGIQUE
Dans cette partie, vous utiliserez une classe documentée pour créer un serveur UDP sur une Raspberry.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL
- Programmation Orientée Objet — Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES
- Utiliser une classe — Niveau 2
- Créer une classe — Niveau 2

Gaëtan GEORGES
Pascal HAGOT
David IERVESE

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

Le TCP RS sur Raspberry

Présentation de l'évolution du système

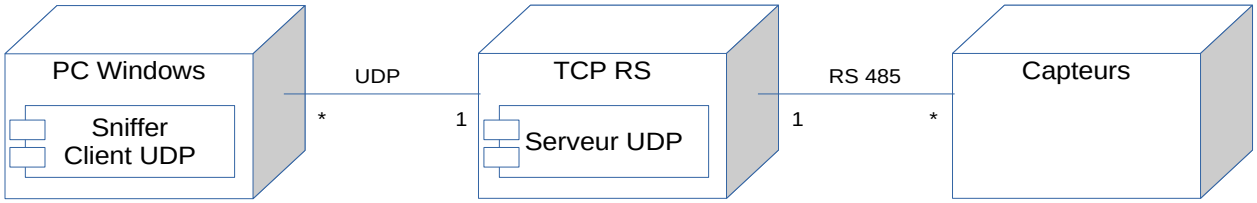
Dans les parties qui vont suivre, vous allez effectuer une mutation de ce projet. En effet, nous souhaitons changer les TCP RS par des Raspberry. Voici au moins 2 raisons :

1. Les TCP RS sont très vite saturés si plusieurs clients font des requêtes simultanément.
2. Les protocoles des TCP RS ne sont pas chiffrés.

Votre mission consistera donc à améliorer l'existant : en utilisant un système plus performant avec chiffrement.

Le diagramme de déploiement

Pour rappel, voici le diagramme de déploiement actuel.



Le TCP RS possède donc un serveur UDP qui reçoit toutes les requêtes des clients UDP (de votre application sniffer par exemple). Lorsqu'il reçoit une requête, le TCP RS la renvoie telle quelle sur le bus RS 485. Les capteurs voient ainsi les requêtes. Si l'un d'eux est concerné (c'est son adresse sur les 2 premiers octets), alors il y répond sur le bus RS 485. Le TCP RS voit la réponse et la transmet au client.

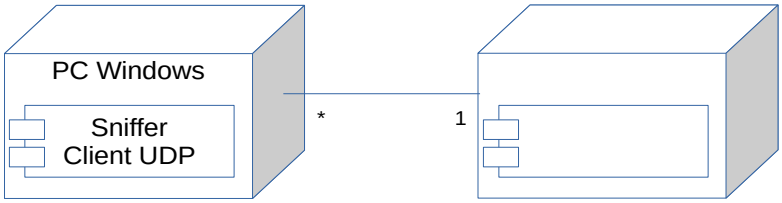
Les phases de l'évolution du système :

- 1ère phase : Mettre en place un serveur UDP sur une Raspberry
- 2ème phase : Chiffrer les données entre le client et le serveur UDP
- 3ème phase : Communiquer avec les capteurs sur le bus RS 485

1ère phase : Mettre en place un serveur UDP sur Raspberry

Dans cette première évolution du système, vous allez mettre en place un serveur UDP sur Raspberry. Il répondra systématiquement à toutes les requêtes de fonction 0x05 provenant des clients.

Compléter le diagramme de déploiement de la 1ère phase en ajoutant les mots : Rapsberry, UDP, Serveur UDP aux bons endroits :



Mise en place d'un projet C++ sur la Raspberry

Sur le répertoire d'échange, récupérer le dossier RaspberryServeurUDP_PourEtudiants

Avec SSHFS-Win-Manager (ou WinSCP), se connecter sur la Raspberry cible avec vos identifiants et coller le dossier que vous venez de récupérer.

Ouvrir les fichiers sources du projet avec VS Code ou un autre éditeur.

Modifier le programme principal pour qu'il affiche « Serveur UDP sur Raspberry ! »

Rappeler la commande nécessaire pour lancer une compilation et le fichier indispensable contenant les commandes de compilation.

Ouvrir le fichier makefile et trouver le nom de l'exécutable qui sera générer lors de la compilation

Compiler et exécuter le programme. Vérifier que le message apparaît.

Création d'un serveur UDP en C++

Pour vous faciliter la mise en place du serveur UDP en C++, nous avons mis au point une classe : IRServeurUDP qui est déjà présente dans le répertoire du projet que vous avez téléchargé. Voici son diagramme UML de classe (Une documentation de cette classe sera accessible ici : http://172.20.21.200/IRClasses)

IRServeurUDP
- m_adresseIPDEcoute : string - m_portDEcoute : unsigned short - m_socketServeurUDP : int - m_nbServeurUDP : int - m_addrServeur : struct sockaddr_in - from : struct sockaddr_in
+ IRServeurUDP() + ~IRServeurUDP() + OuvrirLaSocketDEcoute(portDEcoute : unsigned short, adresseIPDEcoute : string) : bool + EnvoyerUnMessage(message : string) : bool + EnvoyerDesOctets(tableau : const unsigned char *, taille : int) : bool + RecevoirUnMessage(message : string &) : int + RecevoirDesOctets(tableau : unsigned char *, tailleMax : int) : int + FermerLaSocket() : bool

Première version : un serveur UDP qui affiche le message reçu

Le but de cette première version est de faire un serveur UDP qui reçoit un seul message et le renvoie comme un écho. Ensuite, le serveur s'arrête.

> En C++, créer un objet « serveur » de la classe IRServeurUDP.

> En C++, ouvrir la socket d'écoute du serveur sur toutes les adresses IP disponibles de la Raspberry et sur le port 10100 + numéro de poste.

> Expliquer pourquoi vous ne pouvez pas tous avoir le même port d'écoute sur la Raspberry ?

> En C++, ajouter les instructions permettant de récupérer les octets envoyés par le client (votre sniffer) avec la méthode RecevoirDesOctets(). Voir la documentation détaillée de la classe pour vous aider.

> En C++, afficher les octets reçus en hexa.

> En C++, fermer la socket d'écoute.

> Tester votre serveur avec le sniffer des séances précédentes. Avec Wireshark, visualiser les trames circulant sur le réseau entre le client UDP et le serveur UDP.

Deuxième version : un serveur UDP qui répond pour un capteur seulement

Votre serveur n'enverra une réponse qu'à la requête du capteur dont l'id est 0x0821 et si la fonction utilisée vaut 0x05.

> Écrire en C++ cette condition

La réponse devra contenir 5 octets :

- Le premier octet : l'octet de poids fort de l'adresse du capteur
- Le deuxième octet : l'octet de poids faible de l'adresse du capteur
- Le troisième octet : le type de capteur (pour nous ce sont des SP3, voir la documentation)
- Le quatrième : la version (vous indiquerez la version 2.3, voir la documentation)
- Le cinquième octet : ~bcc = ~(octet1 + octet2 + octet3 + octet4)
- Écrire en C++ les instructions permettant de créer un tableau « reponse » contenant les 5 octets précédents.

> En C++, ajouter les instructions permettant de renvoyer les octets reçus. Voir la documentation pour vous aider.

> Enfin, en C++, fermer la socket serveur.

> Tester votre programme avec votre sniffer et vérifier qu'une réponse est envoyée pour le capteur 0x0821.

Troisième version : la boucle infinie

> Ajouter une boucle infinie pour que votre serveur puisse lire les messages et y répondre sans jamais s'arrêter.

> Tester votre programme.