Ilyess NAIT BELKACEM 21404888

Siegfried SAHUC 21502151

Projet Réseau 2020

Dazibao2 :

1) Ce que nous avons réussi à faire :

Choisir l'adresse du serveur sans recompiler le programme.

Maintenir une association avec le serveur.

Recevoir des TVL Notify, y répondre avec des TLV Notify Ack.

Configurer le temps d'expiration des données.

Maintenir une table de données reçues, et les afficher à l'utilisateur.

Publier plusieurs données différentes simultanéments sans relancer le client.

Interface utilisateur : système de menu.

2) Comment cela a été fait :

Choix de l'adresse : l'adresse est demandée au début.

Maintien de l'association : un child envoie un paquet vide toutes les 30 secondes.

Maintenir une table de données et les afficher : un thread s'occupe de maintenir une Glist de la table de données (il attend en permanence d'en recevoir), tandis que le programme principal l'affiche à l'utilisateur quand celui-ci le demande.

La structure :

Nous avons décidé de créer la structure buff qui contient un char* et un size_t afin de simplifier l'envoi des données.

Le traitement des données reçues est opéré grâce aux types, on vérifie buff->buffer+4 pour trouver le type de TLV, et cela nous permet d'afficher les types 6 (warnings), ou d'envoyer un notify ack (type = 3) lorsqu'on reçoit un notify (type = 2).

Les tables de données sont des GList de glib. Nos deux listes sont en variable globales (ainsi la table des données reçues peut être remplie par un thread et lue en même temps par le programme principal).

3) Difficultés rencontrées :

Le choix de structure fut compliqué. Nous avions essayé de créer trois structure, une struct datagramme qui était composée d'une struct header et d'une struct tlv. La structure header était composée de magic, version et body_length; tandis que la structure tlv etait composee de type, length et body. Cette structure complexe fut abandonnée en cours de route, à la faveur de la struct buff et d'aller checker nous même dans buff->buffer+4 le type de donnée reçue.

Une autre difficulté rencontrée fut de faire en sorte de toujours être prêt à recevoir des notify et les ajouter à la table, tout en les affichant et en utilisant le menu. On a pensé faire un fork simple, avec un child qui recevrait en boucle les notify, mais g_list_append retourne une nouvelle liste à chaque fois, aussi cette liste n'avait pas la même valeur chez le processus principal, et nous ne pouvions pas la lire. Nous avons pensé à utiliser un pipe mais celui-ci ne peut transmettre que du texte, aussi aurions-nous pu partager dans un pipe la valeur numérique de l'adresse de la nouvelle glist, mais avons finalement préféré mettre la liste en variable globale et créer un thread.

4) Nos idées non développées :

Nous avons pensé faire le démarrage fiable grâce à des usleep mais n'avons pas eu le temps de nous pencher dessus plus avant.