Mathieu BOISNARD
Valentin FRIES

SNMPC

Documentation technique

control solutions
TERACOM

école supérieure de génie informatique

1. Besoins

L'objectif de la suite logicielle SNMPC est de répondre aux besoins des utilisateurs de contrôleurs Teracom TCW180B & TCW181B-CM. Bien que configurables grâce à une interface Web, cette dernière n'offre que peu de choix à l'utilisateur, n'est pas ergonomique et est unique pour chaque contrôleur.

Cependant, il est possible de les configurer via le réseau grâce au protocole SNMP, implémentant un certain nombre de commandes de contrôle. En plus de cela, nous avons pu développer un dispositif de planification de tâches.

SNMPC est un couple logiciel client/serveur : le programme serveur, hébergé sur un RaspberryPi, permettant une exécution permanente et une liaison sur un réseau local avec les contrôleurs, tandis que les programmes clients se connectent directement au serveur via le protocole TCP/IP.

En plus de sécuriser le dispositif en plaçant un composant logiciel entre l'utilisateur et le matériel, l'utilisation d'un serveur permet à l'utilisateur de s'y connecter depuis n'importe quel poste disposant d'une connexion internet.

2. Fonctionnalités

2.1 Programme Client

Le mot d'ordre du client est la simplicité : disposant d'une interface claire, il permettra à l'utilisateur de modifier les configurations aussi bien des relais que du programme serveur en toute transparence. Ses principales fonctionnalités sont :

2.1.1 Authentification

Le client doit s'authentifier correctement sur le programme serveur pour pouvoir établir une connexion pérenne. Ainsi, avant toute chose, l'utilisateur devra saisir l'adresse IP de la machine hébergeant le serveur, un nom d'utilisateur et un mot de passe valides. Ce couple login/mot de passe est envoyé *hashé* au serveur pour vérification.

Si la réponse du serveur est négative ou si le client ne reçoit aucune réponse après un certain laps de temps, l'utilisateur ne sera pas autorisé à accéder aux autres fonctionnalités et vues du logiciel.

Si la réponse du serveur est positive, la connexion est validée et l'utilisateur est alors autorisé à accéder aux autres fonctionnalités et vues du logiciel.

2.1.2 Gestion du matériel

Une fois correctement identifié, le client reçoit un paquet de la part du serveur contenant la liste des appareils et des informations concernant l'état d'activité de leurs relais. Pour chaque appareil actif, un nouvel onglet de contrôle sera créé dans la fenêtre principale. Si l'un des appareils est injoignable, un nouvel onglet sera tout de même créé pour permettre à l'utilisateur de modifier sa configuration IP ou de tenter une reconnexion.

Il est de plus possible d'ajouter un nouvel appareil à la liste du serveur en précisant son adresse IP, un nom qui lui sera associé, ainsi que ses communautés de lecture et d'écriture. L'utilisateur peut aussi supprimer un appareil de la liste gérée par le serveur.

2.1.3 Configuration des relais et planification

Chaque appui sur un bouton de l'interface graphique déclenche l'exécution d'une fonction qui formate les informations concernées (commande, nouvelle configuration, planification...) avant de les transmettre au serveur avec l'identifiant de l'appareil concerné. L'interface est mise à jour en fonction des réponses du serveur, et les informations sont mises à jour automatiquement.

2.1.4 Sécurité des paquets

Afin de ne pas communiquer en clair sur le réseau, chaque paquet transitant entre l'application cliente et l'application serveur sont chiffrés de manière à les rendre inintelligibles en dehors de ce contexte applicatif.

Ainsi, pour pouvoir communiquer et comprendre les réponses du serveur, le client est capable de chiffrer et déchiffrer correctement les paquets envoyés et reçus. Il va de plus *hasher* le nom d'utilisateur et le mot de passe lors de la connexion pour qu'il soit « impossible » de retrouver les valeurs d'origine.

2.1.5 Interface de visualisation

L'interface graphique joue un rôle important au sein de l'application cliente : elle permet de rendre agréable et ergonomique la visualisation de l'état et des informations des différents relais. En effet, chaque paquet reçu du serveur et contenant des données relatives aux appareils commandés entraîne une mise à jour de l'affichage et des informations contenues dans l'interface.

De cette manière, l'utilisateur a accès de manière aisée et en temps réel à toutes les données concernant ses relais.

2.2 Programme Serveur

Le rôle du serveur est de centraliser les requêtes des clients, de les analyser et de les traiter avant de transmettre des ordres d'exécution aux différents appareils. Il est connecté sur le même réseau local que les appareils tout en étant lui-même accessible via internet, servant à la fois de passerelle et de *pare-feu* protégeant les contrôleurs.

2.2.1 Authentification

Les couples nom d'utilisateur/mot de passe du serveur sont enregistrés dans un fichier XML. Lorsqu'il reçoit des informations de connexion *hashées*, le serveur va les comparer avec le contenu du fichier chargé en mémoire. Ces informations de connexion peuvent être modifiées à tout moment, ce qui entraînera une mise à jour du contenu du fichier XML.

2.2.2 Gestion du matériel

Une fois un client connecté, le serveur émet une requête SNMP à destination de chaque appareils en se basant sur une liste d'adresses IP enregistrée dans un fichier XML. Si l'un d'eux répond positivement, le serveur fait suivre une série de requêtes d'informations et d'état le concernant, mettant à jour ses propres informations avant de les transmettre au client. Si l'un des appareils de la liste est injoignable, le client en est de même notifié.

Comme précisé précédemment, les informations de base des appareils sont enregistrées dans un fichier XML contenant un nom arbitraire, une adresse IP permettant de les localiser sur le réseau local, ainsi que leurs communautés. Il est possible d'éditer ce fichier de manière à ajouter, supprimer ou mettre à jour une entrée concernant l'un des appareils.

2.2.3 Traitement des paquets client et interactions SNMP

Le serveur doit être constamment en écoute de paquets de la part des clients auxquels il est connecté. Il doit être capable d'analyser la demande et d'exécuter les traitements correspondant.

Dans le cas d'une requête d'exécution de commande SNMP, le serveur transmet la commande à l'appareil concerné et reste en attente d'une réponse de ce dernier, qui sera renvoyée au client – changement d'état, message d'erreur... –.

Les paquets clients permettent aussi de commander la planification de tâches, la gestion des appareils, les informations de connexion... En bref, toutes les informations gérées par le serveur.

2.2.4 Sécurité des paquets

À l'instar du programme client, le serveur est capable de communiquer de manière chiffrée avec les clients connectés. Ainsi, chaque paquet envoyé ou reçu passe dans des fonctions de traitement qui renvoient dans un cas une valeur inintelligible, et dans l'autre la réelle signification de l'information. Il est de plus capable de *hasher* le couple nom d'utilisateur/mot de passe pour pouvoir mettre à jour le fichier XML contenant ces informations.

2.2.5 Planificateur

L'un des points forts du programme serveur est sa capacité à planifier des exécutions de commandes SNMP vers les différents appareils configurés. Il est ainsi possible aux clients d'envoyer des requêtes de planification qui sont enregistrées par le serveur dans un fichier XML.

Un fil d'exécution parallèle est chargé de lire le contenu de ce fichier lorsqu'il est modifié et de mettre à jour ses informations pour exécuter les tâches planifiées aux moment souhaités.

De cette manière, même si aucun client n'est connecté, le serveur est capable d'envoyer des requêtes SNMP permettant par exemple l'activation d'appareils qui démarreront certains appareils électriques de la maison avant que l'utilisateur ne soit chez lui.

2.2.6 Mini-Shell

Partie « utilisateur » du serveur, un mini-shell permet de prendre la main directement sur le programme serveur depuis la machine hôte. Ce fil d'exécution parallèle permettra à un administrateur serveur de saisir lui-même des commandes de contrôle qui seront ensuite interprétées et exécutées par le programme.

3. APIs

Chacune des bibliothèques tierces utilisées lors du développement de SNMPC a été choisie soigneusement pour correspondre à nos besoins, se conformer aux *standards* de programmation mais aussi pour leur support multi-plateforme, tout particulièrement GNU/Linux & Microsoft Windows, les deux plateformes cibles de ce projet.

3.1 net-snmp

Le protocole SNMP (*Simple Network Management Protocol*) est un protocole réseau de communication permettant l'administration et la gestion distante de matériel connecté. Embarquant un serveur SNMP intégré, les contrôleurs Teracom peuvent ainsi interpréter des paquets envoyés par des applications tierces.

Intégrée uniquement sur le serveur (le client ne disposant d'aucun moyen de communication directe avec les appareils), la bibliothèque net-snmp permet de faire interagir un programme et un composant matériel. Ainsi, nous avons développé notre propre structure d'abstraction permettant l'envoi et la récupération des valeurs de retour d'une requête SNMP.

```
SNMPRequest *InitRequest(Device*, COMMUNITIES);
char **SendRequest(SNMPRequest**, const char*, char*, char**, unsigned short *);
void CloseRequest(SNMPRequest**);
```

```
typedef struct SNMPRequest
{
    netsnmp_session session;
    netsnmp_session *ss;
    netsnmp_pdu *pdu;
    netsnmp_pdu *response;

    oid anOID[MAX_OID_LEN];
    size_t anOID_len;
    COMMUNITIES cm;

    netsnmp_variable_list *vars;
    struct tree *mib_tree;
    int status;

    unsigned short *device_disconnected;
} SNMPRequest;
```

```
typedef enum {
    COMMUNITY_PUBLIC,
    COMMUNITY_PRIVATE
} COMMUNITIES;
```

3.2 pthread

pthread est la bibliothèque standard POSIX pour le développement multi-threading en C. Une implémentation Windows permet de ne développer qu'une unique gestion portable du multi-processing au sein de nos programmes.

Permettant de dissocier l'interface et la communication réseau sur le client, la gestion multi-processing est une ressource essentielle au bon fonctionnement du serveur : l'écoute de nouveau clients doit être constante, des interfaces d'écoute, d'envoi et de traitement de données propres à chaque client doivent être mises en place, un shell utilisateur doit être accessible... Autant de contraintes de parallélisation qui ne peuvent être satisfaites de manière purement procédurale.

3.3 libxml2

Initialement développée par l'équipe du projet GNOME, cette bibliothèque facilitant la gestion de fichiers XML a très vite été rendue libre et multi-plateforme. Elle est utilisée dans le programme serveur pour l'enregistrement pérenne des données concernant les utilisateurs, devices et tâches planifiées, ainsi que les configurations de l'application.

Chargé en mémoire au démarrage du serveur, le contenu des fichiers XML est mis à jour aussi bien dans les structures que dans le système de fichiers dès que l'une des données concernées est ajoutée, supprimée ou modifiée.

3.4 GTK2.0

Incontournable du développement d'interfaces sur système GNU/Linux, GTK a aussi l'avantage de proposer une API graphique compatible Windows. C'est donc sans grande hésitation que nous avons fait ce choix pour le développement GUI du programme client.

Se reporter au *Manuel Utilisateur* pour plus de détails concernant l'interface graphique du client SNMPC.

3.5 *OS-based* sockets

Composant intimement lié au système d'exploitation, il n'existe pas de réelle bibliothèque multi-plateforme standard de gestion des sockets. Les plateformes cibles de SNMPC étant uniquement GNU/Linux et Windows, nous avons pris la décision d'utiliser les APIs système (winsock2.h pour Windows, sys/socket.h pour Linux), de redéfinir les types de données et créer des fonctions d'abstraction de manipulation de manière à obtenir un développement réseau portable.

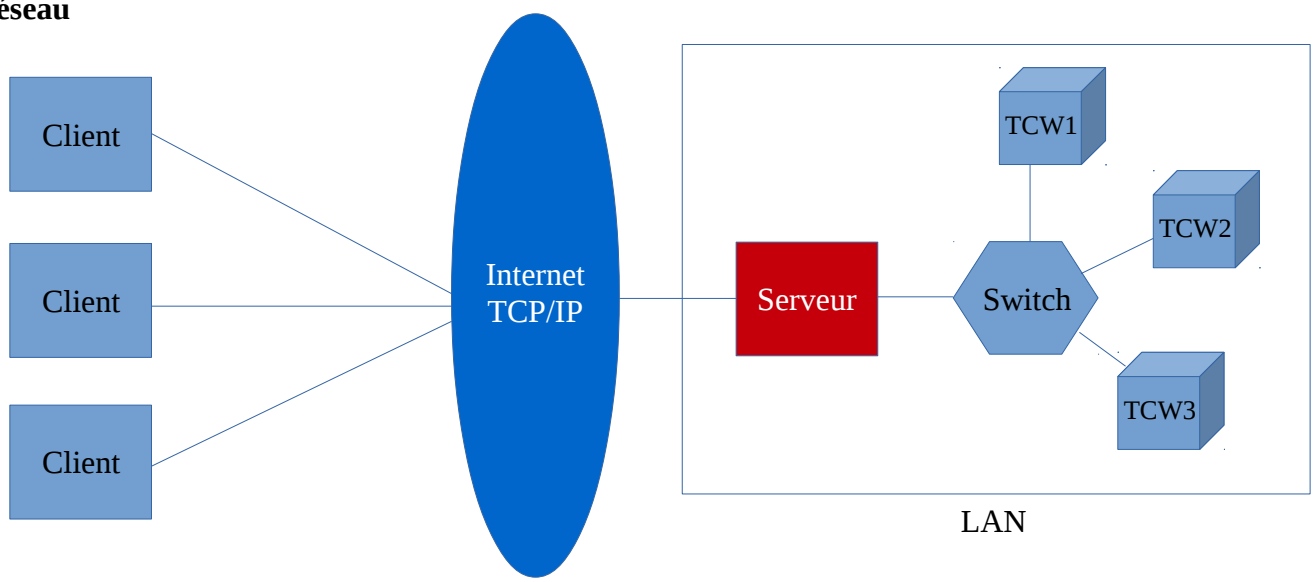
```
#ifdef WIN32
    #include <windows.h>
    #include <winsock2.h>
#elif defined (linux)
    #include <errno.h>
    #include <sys/types.h>
    #include <sys/stat.h>
    #include <sys/socket.h>
    #include <netinet/in.h>
    #include <arpa/inet.h>
    #include <unistd.h>
    #include <netdb.h>
#else
    #error "SNMPC is only developped for Linux & Windows platforms"
#endif
```

```
#if defined (linux)
    typedef int SOCKET;
    typedef struct sockaddr_in SOCKADDR_IN;
    typedef struct sockaddr SOCKADDR;
    typedef struct in_addr IN_ADDR;
    typedef struct hostent HOSTENT;
#endif
```

```
#if defined (linux)
    #define INVALID_SOCKET -1
    #define SOCKET_ERROR -1
    #define closesocket(s) close(s)
```

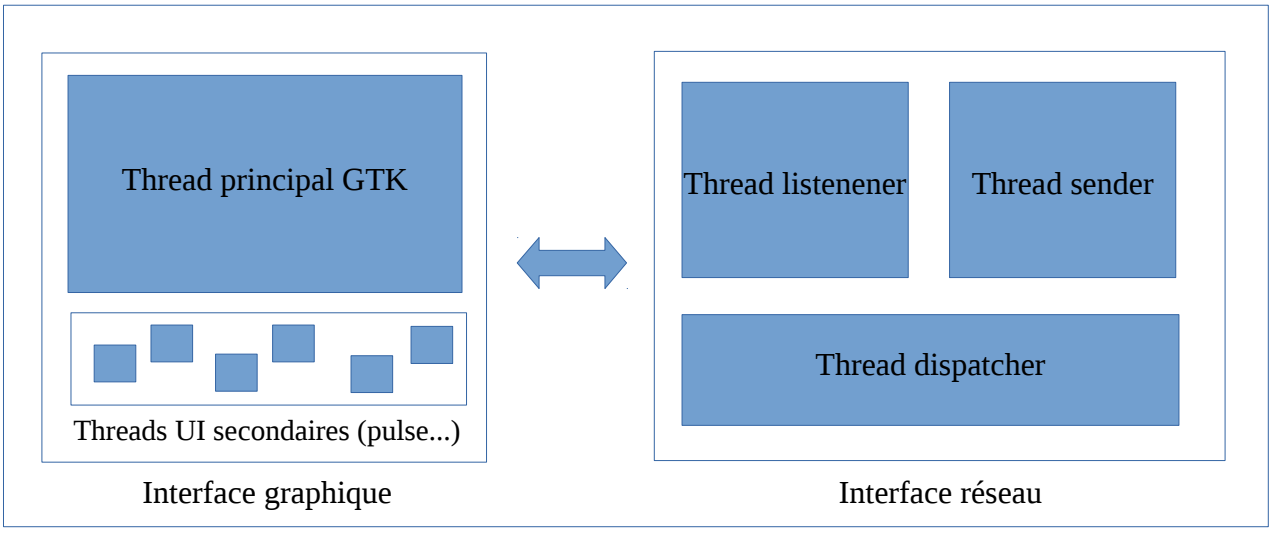
4. Architectures

4.1 Réseau



4.2 Multi-processing

4.2.1 Client



4.2.2 Serveur