A l'attention du jury de soutenance

Rapport de Projet

OpenBox4 - Gnbox

Version 1.0

Année 2010-2011

Auteurs : Maxime Grare, Jean-Charles Vernier

Université du Littoral
50, rue Ferdinand Buisson
BP 699 – 62228 Calais Cedex
Téléphone : 03 21 46 36 00
Télécopie : 03 21 46 36 69

Contrôle du Document

Historique des versions

Date	Version	Auteurs	Principales Modifications
4 Mars 2011	1.0	Grare Maxime, Vernier Jean-Charles	Création du document
25 mars 2011	1.0	Grare Maxime, Vernier Jean-Charles	Finalisation du document

Distribution
Aux auteurs
Au tuteur de projet
A la communauté OpenBox4

Sécurité et confidentialité
Non Applicable

Responsabilité
Non Applicable

Notes sur cette édition
Ras

Table des matières

1 Introduction

1.1 Cadre

Dans le cadre de notre formation de Master Informatique en Ingénierie du Logiciel Libre, un projet tuteuré nous a été attribué. Celui-ci consiste à créer une interface graphique pour une neufbox 4. Ce projet se déroule tout au long de l'année universitaire.

1.2 But

Le but de ce projet est de développer une application qui permet d'administrer une neufbox 4 depuis un ordinateur sans se connecter à l'interface web. Le but est également de participer au projet libre OpenBox4 à travers Gnbox.

1.3 Plan

Dans ce rapport, nous allons présenter le déroulement du projet. Ainsi, vous trouverez dans ce rapport, une présentation du projet OpenBox4, une présentation du projet Gnbox, son fonctionnement et nous terminerons par une conclusion.

2 Présentation du projet OpenBox4

2.1 La neufbox 4

La neufbox est le modem-routeur fourni par le fournisseur d'accès Neuf/SFR. Elle est présente dans 4 millions de foyers en France et est disponible dans n'importe quel supermarché pour environ 50€. La neufbox a été ouverte à la modification et est ainsi devenue un nouveau terrain de jeu pour les adeptes du logiciel libre et les amateurs d'informatique embarquée. La box permet de faire beaucoup de choses en plus de sa fonction principale. En effet, il est possible de faire de l'auto-hébergement, en faire un serveur web ou un serveur FTP, développer des nouveaux mods, etc.

2.2 La communauté

Une communauté s'est formée autour de la neufbox pour collecter des informations au niveau matériel et logiciel et ainsi est né le projet OpenBox4 dont le but est de créer un firmware alternatif pour la neufbox 4. Un wiki est à disposition des personnes qui souhaitent avoir des informations complémentaires (http://www.neufbox4.org/wiki/). Les membres de cette communauté peuvent s'exprimer et émettre leurs idées sur le forum (http://neufbox4.org/forum/) et les news sont affichées sur le blog (http://www.neufbox4.org/blog/). Le forum compte un peu plus de 900 membres et chacun d'eux a la possibilité de participer au projet et d'étendre les fonctionnalités de la neufbox. Certains ont réussi à héberger un site web sur leur propre neufbox (exemple à cette adresse http://fxmx86.mine.nu/) ou encore flasher la box pour y mettre un nouveau mod.

3 Présentation de Gnbox

3.1 Cahier des charges

Le but de ce projet est donc de développer une interface graphique pour l'a neufbox 4. De telles applications existent déjà : l'interface classique accessible via un navigateur web (192.168.1.1 par exemple) ou encore le Moniteur neufbox (http://www.moniteur-neufbox.com/). Gnbox devra pouvoir administrer une neufbox entièrement via l'application comme on le ferait via l'interface classique, tout en étant compatible avec les systèmes d'exploitation les plus courants.

Nous avons commencé par choisir un langage de programmation, nous avons hésité principalement entre C++ et Python, mais au vu de nos expériences, nous avons finalement choisi C++ et la librairie graphique gtkmm. Le projet faisant partie d'OpenBox4, il est bien évidemment libre (licence GPLv2).

Une fois l'architecture de base définie, nous avons isolé toutes les fonctionnalités sur les interfaces existantes pour créer un cahier des charges basique. Nous avons ainsi recensé les fonctionnalités suivantes :

- Accueil
  - Connexion
  - Date et heure
  - Adresse IP internet
  - Version de Firmware
- État
  - État des services
    - Internet
    - Téléphone
    - TV
  - Informations du modem
    - Adresse IP de la box
    - Adresse MAC de la box
    - Durée de fonctionnement de la box
  - Informations de la ligne
    - Débit descendant
    - Débit montant
    - Marge de bruit descendant
    - Marge de bruit montant
    - Affaiblissement descendant
    - Affaiblissement montant

- Mode de transmission (ADSL2+)
- Téléphonie
  - Téléphonie sur IP
    - État de la ligne
    - État du combiné
  - Historique d'appel
    - activé / désactivé
    - affichage des historiques d'appels (si activé)
  - Appels manqués
    - Activé / désactivé
- Réseau
  - Général
    - État des ports connectés à la box
      - TV
      - PC 1
      - PC 2
      - PC 3
      - Wifi (+ nombre de personnes connectées)
    - Liste des clients connectés
      - nom, adresse MAC, adresses IP, mode de connexion
  - WAN
    - Profil
      - Identifiant
      - Mot de passe
  - DNS
    - DNS Local
      - Adresse IP
      - Nom d'hôte
  - DHCP
    - Serveur DHCP
      - Activé / désactivé
      - Adresse de début
      - Adresse de fin
      - Bail (en secondes)
    - Adresses statiques
      - Adresse | Adresses MAC
  - NAT
    - Translation de ports
      - Nom, protocole, type, ports externes, IP destination, port destination, activation

- UPnP
  - activé / désactivé
  - nombre de règles (si activé)
- Route
  - Table de routage
    - Destination, masque de sous-réseau, passerelle
- Filtrage
  - Filtrage
    - Filtrage personnalisé / désactivé
    - options
      - Protéger les ordinateurs Windows de l'internet
      - Autoriser l'envoi de courriels uniquement par l'intermédiaire des serveurs mail du groupe SFR
      - Bloquer le ping entrant

- Wifi
  - Général
    - point d'accès
      - État
      - SSID
      - Diffusion SSID
      - Canal
      - Mode radio
      - Chiffrement
      - Clé
      - Filtrage MAC
    - Poste connectés
      - Adresse IP | Adresse MAC
  - Configuration
    - Configuration générale
      - Activation borne Wifi | activé / désactivé
      - SSID
      - Diffusion du SSID | activé / désactivé
      - Canal
      - Mode radio | auto, 11b, 54g
  - Chiffrement
    - Chiffrement
      - Système
      - Type de clé
      - Clé

- Filtrage MAC
  - Filtrage MAC
    - Activation du filtrage
  - Adresse MAC autorisées
    - Adresses MAC
- Hotspot
  - Général
    - Hotspot
      - État
      - Mode
    - Postes connectés
      - Liste des postes connectés
  - Configuration
    - Configuration
      - Activation
      - État
- Application
  - Général
    - Application
      - Partage de fichiers
      - Partage d'imprimantes
      - Serveur multimédia
  - Partage de fichiers
    - Partage de fichiers Windows
      - Statut du service
      - Activer / désactiver
  - Partage d'imprimantes
    - Partage d'imprimantes
      - État du service
      - Mode bidirectionnel | activé / désactivé
  - Serveur multimédia
    - Service UPnP-AV
      - Statut du service
      - Activation du service UPnP-AV | activé / désactivé
- Maintenance
  - Général
    - Maintenance
      - Protection de l'accès à l'interface
      - Profil ADSL
  - Système
    - Redémarrage

        - Redémarrage de la neufbox
    - Paramètres de configuration
        - Importer
        - Exporter
        - Restaurer
- Administration
    - Administration de la neufbox
        - Protection (désactivée, mot de passe, bouton de service, les 2)
        - Identifiant
        - Mot de passe
        - Confirmation mot de passe
- Personnalisation
- Ligne ADSL
- Diagnostic
- Tests

L'accès devra être protégé via le login/mot de passe de la box.

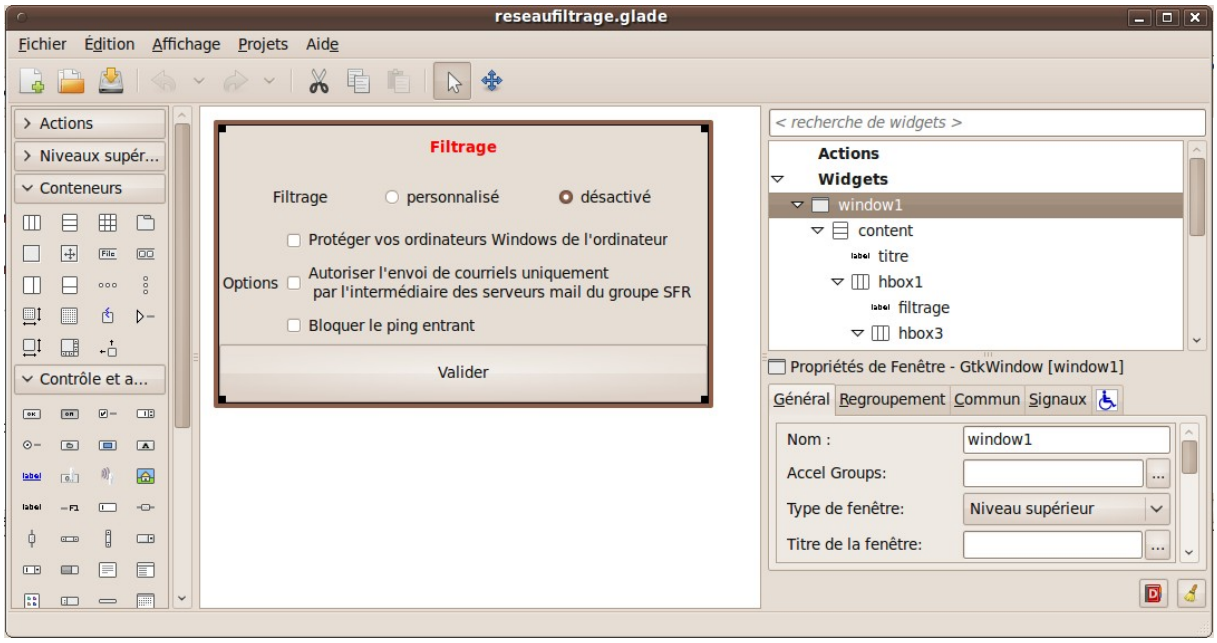
3.2 Les outils utilisés

Comme dit précédemment, nous avons choisi de développer en C++ plutôt qu'en Python. Il y a plusieurs raisons à ce choix. Nous avons voulu choisir un langage que nous connaissions afin de ne pas trop bloquer sur des problèmes simples. De plus, il nous fallait un langage qui soit capable de gérer une interface graphique assez facilement, sachant que c'était la première fois que nous développions une interface graphique lourde. Enfin, nous avions la possibilité de recevoir de l'aide plus facilement en C++ via les professeurs et les étudiants de l'Université.

Concernant le développement de la partie graphique, nous sommes parti naturellement sur gtkmm, qui est en fait l'interface C++ officielle de l'interface graphique GTK+ et GNOME. Ce choix s'est imposé à nous, notamment par le fait que gtkmm est portable sur la plupart des systèmes d'exploitation.

Pour faciliter la création des interfaces graphiques, nous avons opté pour l'utilisation de Glade. C'est une application qui permet de créer son interface en quelques clics et qui génère un fichier .glade, dans un format xml. C++ est ensuite capable d'utiliser ces fichiers pour construire des fenêtres. Cela facilite énormément le développement car on a une idée directe de l'apparence de l'application sans avoir à écrire la moindre ligne

de code, et il est très facile de modifier le fichier glade pour déplacer un objet sans avoir à développer en C++.



Exemple de fenêtre Glade

La communication avec la neufbox se fait par des requêtes HTTP. Nous avons donc cherché une librairie qui permet l'envoi de requêtes simples. Nous avons choisi cURLpp. C'est en fait un wrapper C++ de libcURL, publié sous licence libre (MIT licence).

Pour savoir comment communiquer avec la neufbox, nous allons suivre l'API REST (http://dev.efixo.net/doc/api-rest.html) publiée par la société Efixo. Cette API présente les requêtes à envoyer, l'explication des résultats retournés et de leurs valeurs. En fait, tous les résultats sont au format xml et il suffit de lire ce xml pour connaître la réponse de la neufbox.

Exemple de code xml retourné suite à une requête :

```
<?xml version="1.0" ?>
<rsp stat="ok" version="1.0">
        [resultat]
</rsp>
```

Pour exploiter les résultats xml présentés ci-dessus, nous avions besoin d'un parser xml C++. Il existe de nombreuses librairies qui font cela telles que Xerces-C++ (xerces.apache.org/xerces-c/), TinyXml (www.grinninglizard.com/tinyxml/), expat (expat.sourceforge.net/) ou encore libxml++ (libxmlplusplus.sourceforge.net/). Comme les résultats xml retournés sont très simples, nous avons choisi TinyXml, qui malgré

des performances moindres est beaucoup plus facile à utiliser que ses concurrents.

Afin de faciliter la compilation, nous avons aussi choisi d'utiliser CMake, qui permet de compiler un projet et de générer un exécutable en 2 lignes de commande.

Comme nous faisons partie d'OpenBox4, nous avons rejoint ce projet sur gna.org, une plateforme d'hébergement de logiciel libre. Les sources sont disponibles à l'adresse http://svn.gna.org/viewcvs/openbox4/. Gna fonctionne avec le CVS subversion, et nous l'avons donc utilisé. Nous ne connaissions que git et nous avons donc dû apprendre à l'utiliser.

Enfin, notre projet a également bénéficié d'une page wiki sur openbox4.org (http://www.neufbox4.org/wiki/index.php?title=Gnbox) où nous avons présenté le sujet et son avancement. De plus, nous avons un sujet sur le forum consacré au projet (http://www.neufbox4.org/forum/viewtopic.php?id=1566) sur lequel nous avons pu communiquer avec la communauté de notre avancement et bénéficier à la fois de leurs retours et de leur aide.

4 Fonctionnement

4.1 Apprentissage

Avant de commencer à développer, nous avons eu besoin d'apprendre à se servir de la plupart des outils cités ci-dessus.

Nous avons commencé avec gtkmm. Pour cela, nous avons suivi le tutoriel officiel http://library.gnome.org/devel/gtkmm-tutorial/unstable/. En parallèle, nous avons récupéré un cours de Monsieur RAMAT et de Monsieur QUESNEL. En suivant ces deux sources d'informations, nous avons pu acquérir une base suffisante pour développer l'application.

De plus, nous avons dû apprendre à se servir de subversion que nous ne connaissions pas mais cela s'est fait relativement vite en suivant plusieurs tutoriels, notamment celui fournit par gna.org.

Ensuite, nous nous sommes réparti les choses à apprendre. Maxime a mis en place le système d'onglets et de sous-onglets en se basant sur un tutoriel du site du zéro qu'il a ensuite adapté pour intégrer les sous-onglets. De plus, il a appris à se servir de Glade et surtout à l'intégrer et à l'utiliser dans du code C++.

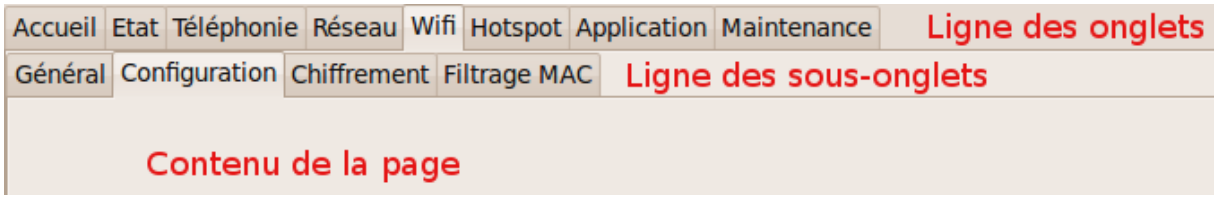
De son coté, Jean-Charles a cherché comment faire des requêtes HTTP et a choisi la librairie cURLpp. Il a ensuite appris à l'utiliser en suivant des exemples trouvés sur internet. Cependant, l'adaptation du code dans notre projet s'est révélé compliqué en raison du manque d'informations et du fait que nous n'avions jamais utilisé ce type de librairie. En effet, nous avons rencontré de gros problèmes pour configurer le CMake avec cURLpp mais Monsieur RAMAT nous a aidé à résoudre notre problème. Une fois cette partie fonctionnelle, Jean-Charles s'est également occupé du parser xml. Encore une fois, nous avons eu beaucoup de mal à trouver un parser xml qui correspondait à nos besoins car la plupart des parsers existants utilisent des fichiers xml alors que dans notre cas, le xml est stocké dans une variable. C'est pour cette raison que nous avons opté pour TinyXml et Jean-Charles a suivi des exemples pour comprendre son fonctionnement.

Une fois ce travail d'acquisition des bases terminé, nous avons partagé nos expériences pour que chacun sache faire ce que l'autre a appris. De cette manière, nous avons gagné du temps et nous avons pu commencé à développer réellement l'application plus rapidement que si nous avions tout fait en binôme.

4.2 L'application Gnbox

Dans cette partie, nous allons détailler le fonctionnement de l'application de manière plus technique.

D'un point de vue graphique, nous avons décidé d'organiser les fonctionnalités avec un système d'onglets et de sous-onglets pour faciliter la navigation dans l'application.



Organisation de l'application

La fenêtre globale qui contient l'application est décrite dans le fichier fenetre.hpp. C'est dans ce fichier qu'on définit les onglets et les sous-onglets. Cette classe est donc commune à toute l'application. A l'inverse des autres classes, fenetre.hpp n'utilise pas de fichier glade. Le rendu de la fenêtre est totalement écrit avec du code C++ et gtkmm.

La partie « Contenu de la page » sera différente pour chaque onglet/sous-onglet. Pour chaque sous-onglet, nous avons en fait une classe qui récupère le fichier glade correspondant et l'exploite pour ensuite l'afficher.

```
builder=Gtk::Builder::create_from_file("glade/etatservice.glad
e","content");
```

Cette ligne de code va récupérer le conteneur « content » du fichier « etatservice.glade ». Il suffira ensuite d'afficher ce qu'on a récupéré dans l'application.

Outre cette partie graphique, cette classe est également chargée de communiquer avec la neufbox via la librairie cURLpp.

```
curlpp::Cleanup myCleanup;
curlpp::Easy myRequest;
myRequest.setOpt<Url>("http://neufbox/api/1.0/?
method=hotspot.getClientList\&token=b4b6a05440d6d1c1e0da69321d
bb3fb83a2b2ffb");
myRequest.perform();
```

Ces lignes de code sont un exemple très simple d'utilisation de cURLpp. Le « cleanup » gère automatiquement les initialisations/destructions des instances de cURLpp. La variable

« myRequest » correspond à la requête HTTP à envoyer. Ici, on appelle une fonction de la neufbox et cet appel est effectué réellement dans le « perform ». De plus, « perform » va afficher le résultat xml dans le terminal (action par défaut).

On va donc récupérer le résultat en xml. Ce résultat sera ensuite « parsé » avec TinyXml et l'affichage sera adapté en fonction de la réponse de la neufbox. Imaginons que l'on récupère la liste des clients connectés à la box, on va mettre à jour le tableau de l'interface prévu pour l'affichage des clients. C'est avec ce système qu'on va pouvoir interagir avec la neufbox.

5 Conclusion

5.1 Problèmes rencontrés

Le premier problème rencontré a été le choix du langage à utiliser. Ce n'était pas vraiment un problème en soi, mais ce choix était important car il allait avoir de grosses conséquences sur l'évolution du projet. Après concertation avec Monsieur RAMAT, nous avons opté pour le C++ car cela nous permettait de nous perfectionner dans ce langage et il nous serait également plus facile d'avoir de l'aide en cas de problèmes.

Le second problème a été l'utilisation des Autotools (Autoconf et Automake) qui permettent la configuration automatique du code source ainsi que la génération des makefiles. Nous voulions utiliser ces outils pour notre projet mais nous n'avons pas réussi à les mettre en place et nous ne voulions pas perdre trop de temps pour les faire fonctionner correctement. C'est pourquoi nous avons décidé d'utiliser CMake qui permet de générer automatiquement des makefiles en utilisant les fichiers de configuration CMakeLists. Nous avons ainsi évité de perdre trop de temps à mettre en place les Autotools, de plus nous étions un peu familier avec CMake car nous avons eu l'occasion de l'utiliser en cours.

Cependant, nous avons rencontrés des problèmes pour écrire les fichiers CMakeLists. En effet, il n'était pas évident de trouver les bonnes commandes à intégrer aux fichiers pour spécifier les librairies que nous utilisons (gtkmm, cURLpp, ...) dans le projet.

Comme énoncé précédemment, nous ne connaissions pas gtkmm, nous avons donc appris à l'utiliser ensemble en faisant quelques tutoriels. Mais pour ne pas perdre temps pour la suite du projet, nous avons décidé de nous répartir les tâches pour avancer plus rapidement. Selon nous, cette répartition a été bénéfique car elle nous a permit de gagner beaucoup de temps au niveau du développement du projet.

5.2 Avancement

A l'heure où nous écrivons ce rapport, l'application n'est pas encore fonctionnelle. Pour le moment, l'interface en elle-même est terminée mais il reste à intégrer des fichiers glade. Cependant, comme cela a déjà été fait dans d'autres parties de Gnbox, cela devrait prendre peu de temps.

Par la suite, nous devrons appeler les bonnes requêtes HTTP aux bons endroits dans l'interface, parser les résultats et adapter l'affichage. Mais encore une fois, comme cela a déjà été fait dans des fichiers tests, il

ne nous reste qu'à adapter le code et cela devrait être fait relativement vite.

Le seul éventuel problème que nous pourrions rencontrer concerne le lien entre les signaux à appeler (équivalent des events) et les signaux déclenchés par glade. Nous n'avons pas encore eu l'occasion de regarder comment cela fonctionne exactement et il est possible que nous y passions du temps.

5.3 Bilan

En ce qui nous concerne, nous pensons avoir beaucoup appris grâce à ce projet. En effet, la plupart des outils que nous avons utilisé était pour nous nouveau, et bien que nous ayons passé du temps à apprendre à les maîtriser, on peut considérer que nous savons maintenant s'en servir.

De plus, pour nous deux, c'était la première fois que nous travaillions sur un projet libre et nous avons également appris sur les outils nécessaires à la vie d'un projet OpenSource comme subversion et gna.org.

Si le projet était à refaire, nous changerions principalement deux choses. La première est notre activité au sein de la communauté. En effet, nous n'avons pas énormément participé au forum et quand nous étions bloqués, nous n'avions pas le réflexe de poster notre problème pour avoir une réponse et se faire aider. Par le suite du projet, nous pensons demander plus souvent des retours aux membres de la communauté, principalement pour tester l'application. D'un point de vue plus technique, l'autre gros point noir concerne la portabilité de l'application. Nous avons tous les deux travaillé sous Ubuntu et nous n'avons pas du tout testé le rendu de Gnbox sous d'autres systèmes d'exploitation.

De manière générale, nous sommes satisfaits de notre travail et nous avons beaucoup aimé travailler sur ce sujet. Nous aimerions également remercier Raphäel HUCK et Eric RAMAT pour leur disponibilité et leur aide tout au long du projet.