ZRTC

Analyse fonctionnelle et Etat de l'art

[Sous-titre du document]

Raphaël Paoloni
Foulques de Saint Laon
Gauthier Parant
09/03/2014

Le projet présenté ici consiste en un développement d'un écosystème de type IRC, c'est-à-dire un réseau basé sur un protocole de communication en salon (chat) qui permet à ses utilisateurs de faire bien plus que discuter. Le but serait ainsi d'obtenir un concurrent viable face à IRC, notamment du point de vue des performances.

Ce document a pour but de présenter les possibilités attendues du projet, les technologies actuellement utilisables pour arriver à un tel résultat, ainsi qu'une justification du choix technologique effectué pour le développement.

I. Analyse fonctionnelle

La description des attentes du projet peut se faire ici en 2 parties. La première concernant les actions possibles d'un client, la deuxième le fonctionnement interne de l'application serveur. Les listes suivantes sont non exhaustives. Certaines fonctionnalités pourront être rajouté en cours de développement, de même que certaines ne seront mises en place que si le temps le permet (*)

A. Côté client

Les fonctionnalités suivantes sont attendues pour un client :

- Communication dans un salon avec d'autres clients
- Communication pair à pair avec un autre client unique
- Chiffrement des communications (par signature ou cryptage)
- Partage de fichier
- Streaming vidéo & audio pair à pair *

B. Côté serveur

Les fonctionnalités suivantes sont attendues pour un serveur :

- Gestion des salons de communication (connexion, kick, ban, nickname,…)
- Scalabilité
- Modularité
- Mise en relation de 2 clients (pour la communication pair à pair)
- Chiffrement des communications (signature pour le cas serveur-serveur et serveur-client, cryptage pour le cas client-client)
- Performance (par rapport aux solutions actuelles)

II. Etat de l'art

L'état de l'art consiste en une étude des différents protocoles et moyens techniques existant pour développer le projet. L'objectif ici est de développer un logiciel libre, il ne sera donc pas fait cas des protocoles non libres (ex : Skype) et des protocoles en cour de fermeture (ex : Google talk, lequel s'appuyait sur xmpp mais ne supportera plus nécessairement ce protocole avec le passage à Hangouts).

C. IRC

Le projet étant proche de ce qui est implémenté par IRC, il est normal que ce protocole puisse convenir. Après tout, IRC est un (très) vieux protocole d'internet, standardisé en 1993 et mis à jour en 2000. Il est donc ancien et éprouvé, et ayant été très populaire, il existe un grand nombre de client et de serveurs pour de nombreux systèmes d'exploitations.

De plus son organisation en réseaux de serveurs et la possibilité de parler à des gens n'étant pas sur le même serveur le rend capable de servir un grand nombre d'utilisateurs. Conformément aux objectifs du projet, IRC prend en compte le chiffrement (via la norme TLS comme http) et enfin il gère les communications pair à pair permettant ainsi de parler sans passer par le serveur ou de transférer des fichiers entre utilisateurs.

Malheureusement il ne prend pas en compte les extensions c'est à dire que, par exemple, rien n'est prévu si on veut faire une visioconférence entre deux utilisateurs ou plus. L'autre problème de cette solution est que de par sa nécessité d'utiliser un client spécifique (les clients web pour IRC existent mais restent très pauvres), l'absence de publicité et le design souvent très spartiate des clients, ce réseau est sur le déclin par rapport à des messageries propriétaire comme Skype ou Facebook Messenger.

D. XMPP

C'est le deuxième grand protocole pour faire des salons de discussion. Il possède une autre appellation « Jabber ». Il est bien plus récent (standardisé en 2004). Il est extensible. Il ressemble beaucoup au mail pour la gestion des comptes dans le sens où il utilise le DNS pour la gestion des noms de comptes (on retrouve des adresses de même forme que les adresses mail, liées à un nom de domaine) et que les serveurs se parlent entre eux. En effet quand on cherche à parler à quelqu'un le serveur auquel on est connecté va aller chercher, via le DNS, l'adresse du serveur auquel peut être connecté la personne pour lui parler. Enfin ce protocole prend en compte le chiffrement (via openPGP notamment) ce qui en fait un protocole sécurisé.

E. Chatbox

Il en consiste en de petits salons de discussions et est basé sur les technologies web avec un client en JavaScript et un serveur souvent en PHP. Cette technologie utilise Ajax pour la mise à jour de l'affichage mais souffre de très graves défauts. En effet elle est peu ou pas scalable (une chatbox ne peut généralement pas tenir plus de quelque dizaines ou une petite centaine d'utilisateur), les mises à jour ne sont pas en temps réel mais tout les X temps (limitations imposé par l'utilisation du protocole HTTP qui, jusqu'à très récemment ne permettait pas au serveur de pousser une mise à jour vers le client).

Cette méthode garde tout de même les avantages de HTTP au niveau de la sécurité (si la chatbox utilise HTTPS) et a pour énorme avantage d'être très facile à mettre en œuvre coté serveur et de ne rien demander d'autre qu'un navigateur web coté client

F. webRTC

Une autre solution de communication utilisant les navigateurs web. Cette technologie, très prometteuse permet de faire passer du son (chat vocal), de la vidéo ou n'importe quel type de données entre des navigateurs web. Malheureusement, cette technologie a un problème qui l'élimine immédiatement pour nous : elle est trop récente.

En effet elle n'est pas encore implémenté dans tous les navigateurs (actuellement chrome canary, la version de développement de chrome, l'implémente entièrement), les implémentations sont souvent incompatibles entre elles (bien que de gros travaux aient lieux entre Mozilla et Google pour améliorer cette interopérabilité). C'est donc une technologie à surveiller mais qui n'est pas encore mature.

G. Socket

Le développement depuis la base avec utilisation des sockets permettrait de faire ce que l'on veut et donc de spécifier le protocole utilisé de A à Z. Malheureusement au vu des objectifs du projet et du temps disponibles pour le terminer, soit le développement ne serait pas suffisamment complet, soit les performances de l'application seraient très mauvaises. Dans tous les cas les performances seraient limitées par notre niveau de connaissances dans le domaine de la gestion d'applications en réseau et le découpage des tâches pour les optimiser.

H. 0MQ (zéroMQ)

C'est une bibliothèque libre (LGPL). Elle annonce des performances très élevées, et elle permet de faire facilement des listes de diffusions. Une autre de ses qualités est qu'elle est disponible dans de nombreux langages de programmation, elle permettrait donc une évolution de l'application (modularité) plus facile.

Son inconvénient est d'utiliser des messages et d'avoir son propre protocole, néanmoins cela ne devrait pas déranger lors du développement de l'application. Cela nécessiterait simplement une plus grande attention lors du développement de la communication entre le client et le serveur.

Conclusion

A partir de cette étude, on peut remarquer qu'il est donc plus intéressant et plus pratique d'utiliser le framework réseaux 0MQ comme base pour l'application. Ce dernier permettant l'utilisation de nombreux langages de programmation, il a été décidé au vu de nos connaissances d'utiliser Java au niveau du serveur. Bien qu'il ne soit pas optimal (le C ou C++ aurait été préférable), il garde des performances honnêtes et permettra un développement plus rapide.

Du côté client le choix du langage de programmation se porte aussi vers le Java. D'une part pour les même raisons que précédemment, d'autre part parce qu'il est multiplateformes, il n'y aura donc pas à se soucier des environnements d'exécution.