DÉPARTEMENT D'INFORMATIQUE
Faculté des sciences
Université de Sherbrooke

Projet

Par

GABRIEL HORBATUK 11 104 119
MARC RODRIGUE POULIN 10 232 152

Projet présenté à
GABRIEL GIRARD

dans le cadre du cours
IFT630
Processus concurrents et parallélisme

Sherbrooke
23 avril 2013

Table des matières

1. Description de projet

Notre projet est un système client-serveur permettant de communiquer par la voix avec les autres clients connectés. Les clients connectés peuvent se parler s'ils sont dans la même salle de discussion. Il est possible de configurer le serveur pour ajouter et supprimer des salles de discussion.

Les applications sont codées en C# et le client utilise la librairie Naudio pour gérer tout ce qui a rapport au son.

Nous nous sommes inspirés des applications comme TeamSpeak et Ventrillo pour ce projet. Évidemment, notre système est beaucoup plus simple et comporte beaucoup moins de fonctionnalités.

2. Fonctionnement de l'application

2.1. Le client

L'interface du client est très simple. Pour commencer, le client doit se connecter à un serveur. Il entre l'adresse IP et le numéro de port du serveur. Le client envoie ensuite une annonce au serveur et le serveur lui répond en lui envoyant la liste des salles. Si le serveur ne répond pas, le client abandonne la connexion. Le client peut ensuite commencer à parler. Pour parler, l'utilisateur doit maintenir la touche shift enfoncée.

Nous avons ajouté un formulaire de « Préférences » pour sélectionner quels périphériques à utiliser. On le sélectionne simplement avec des listes déroulantes.

Le client a deux fils d'exécution. On utilise un fil d'exécution pour écouter sur le réseau avec un socket. L'autre sert à envoyer des données captées par le micro au serveur.

Le client peut détecter une perte de connexion avec le serveur à l'aide de temporisateurs. Le client envoie périodiquement un message type « ping » au serveur. Si la réponse du serveur ne vient pas, le client ferme les sockets et met fin à la connexion.

Une fois la connexion établie avec le serveur, le client peut changer de salle de discussion en double cliquant sur le nom de la salle dans la liste de salle, à gauche. Un message est envoyé au serveur pour indiquer que le client a changé de salle.

2.2. Le serveur

Après avoir démarré l'application serveur, il faut mettre le serveur en service en cliquant sur Connexion → Démarrer, dans la barre de menu.

Une fois démarré, le serveur utilise un fil d'exécution pour écouter sur le réseau. Lorsqu'il reçoit datagramme, il lit le premier octet et fait la bonne action selon le type du message reçu.

1 - Connexion

Si le premier octet contient la valeur 1, cela veut dire qu'un client s'est connecté. Le serveur crée donc une connexion pour le nouvel utilisateur et démarre un nouveau fil d'exécution. Ce fil d'exécution servira à envoyer des données à ce client. Une fois le fil d'exécution créé, on l'utilise pour envoyer la liste des salles de discussion au client.

2 – Données du micro

La valeur 2 comme premier octet indique que le datagramme contient des données provenant d'un micro. On doit envoyer ces données à tous les usagers qui sont dans la même salle que l'émetteur. On trouve l'usager source en cherchant parmi la liste des usagers l'usager ayant l'adresse IP de l'émetteur du datagramme. On diffuse ensuite les données parmi les usagers dans la salle associée à l'usager émetteur.

3- Demande de changement de salle

Un premier octet avec la valeur 3 informe le serveur d'une demande de changement de salle par un usager. Le reste du datagramme contient le nom de la salle dont l'usager veut aller. On trouve l'usager source avec l'adresse source du datagramme et on change la valeur de la salle associée à l'usager. Le client n'attend pas de confirmation du serveur.

4 –Déconnexion du client

La valeur 4 indique que l'usager se déconnecte du système. Le serveur libère alors les ressources pour l'usager qui a envoyé la demande.

5 – Ping

La valeur 5 est associée à un ping du client. Lorsque le serveur reçoit un ping, il répond au client en envoyant un datagramme contenant la valeur 5.

Tout comme les clients, le serveur peut détecter qu'un client a perdu la connexion avec un temporisateur. S'il ne reçoit plus les messages de type ping d'un client en particulier, il ferme la connexion du client.

Sur le serveur, il est possible d'ajouter et de supprimer des salles de discussion, mais seulement lorsque le serveur n'est pas en marche. Une interface graphique très simple permet d'accomplir cela. Nous utilisons un fichier nommé config.ini pour garder une liste des salles.

Lorsque le serveur est arrêté (Connexion → Arrêter, dans la barre de menu), il envoie un message à tous les clients connectés pour annoncer sa fermeture. Il envoie la valeur 4 aux clients. Le serveur ferme ensuite toutes les connexions et termine. Les clients, qui ont reçu le message de fermeture, ferment leurs sockets et mettent fin à connexion.

3. Gestion des fils d'exécutions

Pour gérer les fils d'exécutions, nous avons utilisé la classe *ThreadPool* offerte par le Framework .NET. Un Thread pool est un ensemble de fils d'exécution qui sont utilisé pour faire des tâches en arrière-plan. L'utilisation d'un thread pool a grandement facilité la gestion des threads, étant donné que les fils d'exécutions sont gérés automatiquement par le système.

Nous avons aussi utilisé les locks pour assurer l'exclusion mutuelle pour certaines variables. Un lock fonctionne selon le même principe qu'un mutex.

4. Améliorations possibles

Le système peut être amélioré de diverses façons. Une fonctionnalité intéressante serait de pouvoir être en mesure de communiquer par clavardage en plus de la voix.

De plus, la performance du serveur laisse à désirer. Le serveur consomme énormément de temps d'UCT, tellement que nous n'avons pas été en mesure de supporter qu'un petit nombre d'usagers sur même serveur.

Il serait aussi intéressant de pouvoir faire afficher les noms des participants connectés sur les clients.

Une autre amélioration serait de permettre à l'administrateur du serveur de pouvoir créer des salles de discussion pendant que le serveur est en service.