Vue d’ensemble

Devant apprendre à utiliser les librairies de communication TCP/IP de Delphi 5 vu qu’elles pouvaient servir pour créer un système de chat. J’ai donc tenté d’en créer un, de fil en aiguille, j’ai rapidement remarqué que les possibilités étaient très grande, suffisament pour en faire une partie de mon projet final.

J’ai donc créé deux exécutables, le premier servant de serveur auquel plusieurs clients peuvent se connecter, il garde en mémoire les adresses et les noms d’usagers. Il réagit à certaines commandes, permet d’envoyer des messages admin et système, renvoi les messages des clients vers les autres clients et affiche les logs.

Le deuxième est le client, il permet de choisir l’IP et le Port auquel il se connecte. L’utilisateur peut changer son nom, soit avec le bouton ou en ligne de commande. Ce nom reste en mémoire tant que le serveur est ouvert (le formulaire et non la connection). Le client peut envoyer des messages ou des commandes.

Connection TCP/IP

J’ai utilisé l’outil de base de Delphi pour la communication TCP/IP, il s’agit d’un module qui se sert de WinSock. Bien que relativement basique au niveau des fonctionnalités et des performances, il convient parfaitement aux besoins de ce projet.

L’un de ses avantages est qu’il permet le transfert direct de string, il est donc assez simple de transmettre des messages de longueure variable. L’alternative serait d’envoyer les données dans un buffer et le transférer entre le serveur et le client par le stream.

Cet outil permet aussi de faire les transferts en mode “blocking” qui est couramment utilisé dans les applications de streaming.

Le serveur

Le serveur est de loin l’élément le plus complexe du projet. Il doit recevoir et gérer toutes les connexions, répéter les messages à tous les clients, gérer la liste des utilisateurs en répondre au commande.

La répétition des messages, tout comme l’envoi des messages à tous est réalisé grâce à la procédure BroadCast. Cette procédure envoi le message vers tous les clients connectés et l’affiche dans l’écran du serveur. Il ajoute le nom ou l’IP de l’utilisateur ayant envoyé le message.

Lorsqu’il reçoit un message, le serveur le décortique pour voir si il s’agit d’une commande. Les commandes ont le format suivant: cmd####. Dépendamment de la commande utilisée, le serveur réagira différemment. Il y a présentement 5 fonctions:

1. 0000 - Broadcast “Hello World”
2. 0001 - Broadcast “I rulez”
3. 0101 - Afficher l’IP du client
4. 9172 - Changer de nom d’utilisateur
5. 9999 - Afficher l’aide

L’outil WinSock gère automatiquement les connexions en leur assignant un handle (réutilisable) et un index (ActiveConnections[x]).

La liste d’utilisateurs a demandé un peu plus de planification. Le choix final est un array en deux dimensions dynamique (Array of Array of String en Delphi) dont le premier champ est l’IP et le deuxième est le nom. L’array et sa taille sont globals puisque plusieurs procédures et fonctions les utilisent. Il grossit avant chaque ajout.

L’affichage du nom d’utilisateur est géré par la fonction CheckUserList qui retourne soit l’IP, soit le nom lié à cet IP si il y en a un.

Tant que l’utilisateur ne choisit pas de nom, soit en envoyant la commande 9172 ou en appuyant sur le bouton “Register” son IP n’est pas dans la liste. La fonction retourne donc son IP.

Lorsqu’il s’enregistre, l’IP de l’utilisateur et le nom choisi sont envoyés dans l’array. Ils y restent tant que le serveur est ouvert. Donc si l’utilisateur se déconnecte et revient à partir du même IP il gardera son nom.

Il peut aussi changer son nom avec la commande 9172, l’outils de gestion de la liste cherchera si l’IP est déjà dans la liste et modifiera le nom s’il y a lieu.

Le formulaire de serveur est assez épuré, il est en mode sombre pour faciliter la lecture le soir. Il comporte 6 contrôles visibles, 3 boutons, 2 edit box et un memo box.

Le bouton Connect active le serveur sur le port choisi par l’administrateur dans l’edit box prévu à cette effet (celui du haut). Tant que le serveur n’est pas actif, les boutons de déconnexion et d’envoi sont désactivés. Sans surprise le bouton Disconnect coupe la connexion et le Send envoie le message de l’edit box du bas vers les clients. Le memo box sert d’écran d’affichage pour les messages serveur. Les connexions, déconnexions, commandes utilisées et messages des clients y sont affichés en ordre chronologique.

Le Client

L’exécutable client est, en apparence, très simple. Il permet de se connecter à un IP spécifique, d’envoyer des messages, notification visuelle lors de la réception de nouveau message et d’afficher les réponses.

Il a subi plusieurs modifications à la demande des testeurs. Des fonctions de mode noirceure, modification de la taille de la police et modification du nom d’utilisateur ont été ajoutées.

Les boutons doivent être désactivés ou activés en fonction de l’état du client (connecté ou non).

Les couleurs de l’écran d’affichage changent lors de la réception de message, ces couleurs restent jusqu’à ce que la souris se déplace sur le formulaire. Elles retournent aux couleurs d’affichage courant (nuit ou jour).

Le bouton Register se désactive après la première utilisation, l’utilisateur doit donc envoyer une ligne de commande pour changer de nom une deuxième fois.

L’interface est un peu plus chargée que celle du serveur, elle est composée de 12 contrôles visibles et actifs.

Il y a les edit boxes pour définir l’IP, le port, le nom d’utilisateur et le message à transmettre. Les boutons pour faire ces opérations, changer le mode d’affichage et modifier l’affichage. Comme pour le serveur, un memo box est utilisé pour afficher les messages.

La grosseur de la police du texte d’affichage peut être modifiée, de 5 à 20 points.

Les fonctionnalités sont assez simples, il suffit de définir les propriétés de l’objet TWinSocket et d’activer la connexion. Les messages sont reçus et envoyés sous forme de string avec une procédure intégré à l’objet TWinSocket.

Les autres fonctions ne font que modifier les propriétés des différents contrôles.

Conclusion

Le système de chat fonctionne sur un réseau local tant que les ports sont ouverts. La méthode la plus fiable sur le réseau de la classe est de lancer le serveur dans une VM. Des clients sur n’importe quel hôtes ou Vm pourra s’y connecter.

Il y a plusieurs possibilités pour développer cet outil. Les fonctions de la librairie de base permettent le transfert de fichiers et le streaming. D’autres libraries plus complètes comme Indy offrent encore plus de possibilités. L’option qui m’intéresse le plus serait la connexion inter-réseau.

Cette partie de projet m’a permis de me familiariser davantage avec la communication TCP/IP, le langage Delphi et la relation client/serveur. J’ai aussi dû répondre à des demandes de clients désirants plus de fonctions.

C’est définitivement un projet que je vais continuer de développer.

Le code source se trouve sur GitHub : <https://github.com/DLetourneau/DoomChat>