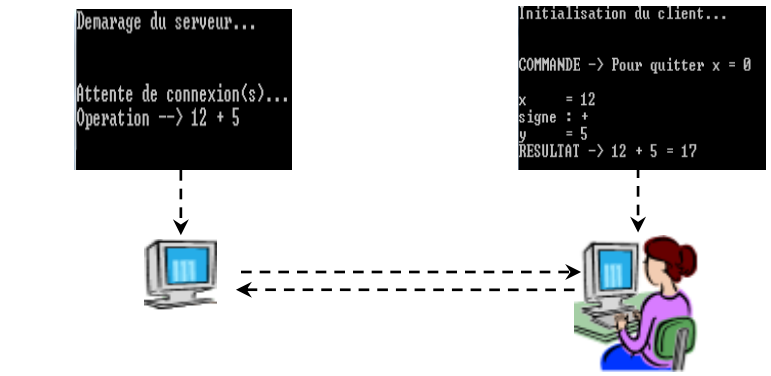
Sockets – TP2 –

I/ Objectif du TP

Le but de ce TP est de faire une réaliser une calculatrice sous une architecture **Client /Serveur**. L'interaction entre le client et le serveur est réalisée par sockets.

I.1- Communication Client/Serveur par Sockets

- Considérant deux machines (Client et Serveur) reliées par un réseau Wifi.
- On se propose de réaliser une calculatrice Windows Standard avec une architecture **Client / Serveur**.
- La machine cliente, envoie une requête pour l'exécution d'une opération de base que le Serveur intercepte. Ce dernier réalise le traitement nécessaire et renvoie le résultat au Client.
- La communication entre le Client et le Serveur se fait par **sockets UDP**.
- Le client et le serveur sont tous deux des applications console.
- La communication entre le Client et le Serveur est réalisée au départ sur une seule machine ensuite sur deux machines physiques.

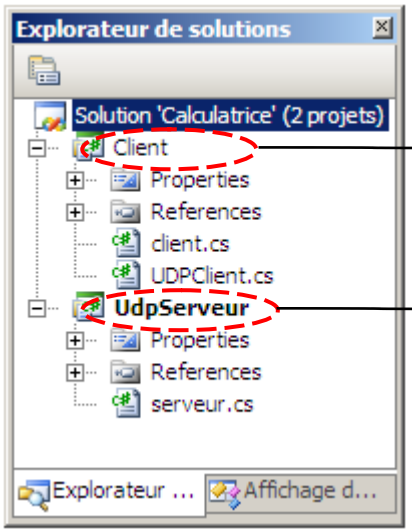


UdpServeur : Application Console

Client : Application Console

Sockets – TP2 –

I.2- Exemple : Réalisation d'une calculatrice avec socket UDP



1. Projet `Client`: contient deux classes **client** et **UDPClient**.

2. Projet `Serveur`: contient une classe **serveur** qui implémente les méthodes invoquées par le client.

1. Projet `Client`:

1. Classe client:

```
using System;
using System.Net;
using System.Net.Sockets;

namespace Client
{
    class client
    {
        private int clientPort = -1;            //port du client
        private int serveurPort = -1;           //port du serveur
        private string serveurHost = null;      //adresse du serveur

        UdpClient Client = null;                //le client

        //constructeur
        public client(int clientPort, string serveurHost, int serveurPort)
        {
            this.clientPort = clientPort;
            this.serveurHost = serveurHost;
            this.serveurPort = serveurPort;
            this.Client = new UdpClient(clientPort);
        }

        public void Close() { this.Client.Close(); }
        public bool Execute(string command, ref string resultat)
```

Sockets – TP2 –

```
        {
            bool ok = true;
            //commande a envoyer
            string req = "127.0.0.1:" + this.clientPort + ":" + command;

            //conversion de la commande en byte
            byte[] rq = System.Text.Encoding.ASCII.GetBytes(req.ToCharArray());

            //envoi de la commande
            Client.Send(rq, rq.Length, this.serveurHost, this.serveurPort);

            //variable de reception du resultat
            IPEndPoint ip = null;
            //reception
            byte[] r = Client.Receive(ref ip);

            //conversion
            resultat = System.Text.Encoding.ASCII.GetString(r);

            ok = !resultat.Equals(0);
            return ok;
        }
    } }
```

2. Classe UDPClient:

```
using System;
using System.Net;
using System.Net.Sockets;

namespace Client
{
class UDPClient
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Initialisation du client...");
            client proc = new client(8081, "127.0.0.1", 8080);
            string x, y, signe, command, resultat = "";

            while (true)
            {
                Console.WriteLine(" ");
                Console.WriteLine("COMMANDE -> Pour quitter x = 0\n");

                //entrer x
                Console.Write("x     = "); x = Console.ReadLine();

                //terminer si x = 0
                if (Int32.Parse(x) == 0) break;
```

Sockets – TP2 –

```
                //entrer le signe
                Console.Write("signe : "); signe = Console.ReadLine();

                //entrer y
                Console.Write("y     = "); y = Console.ReadLine();

                command = x + ":" + signe + ":" + y;

                proc.Execute(command, ref resultat);
                Console.WriteLine("RESULTAT -> " + resultat);
            }
        }
    }
}
```

2. Projet UdpServeur :

1. Classe serveur :

```
using System;
using System.Net;
using System.Net.Sockets;

namespace UdpServeur
{
    class serveur
    {
        static void Main(string[] args)
        {
            int P = 8080;     //port de connexion
            IPEndPoint ip = null;

            //Demarage su serveur
            Console.WriteLine("Demarage du serveur...");
            UdpClient serveur = new UdpClient(P);

            bool loop = true;

            while (loop)
            {
                //Attente de connexion
                Console.WriteLine("Attente de connexion(s)...");
                byte[] tmp = serveur.Receive(ref ip);

                //Conversion des bytes en string (chaine de caractere)
                string data = new System.Text.ASCIIEncoding().GetString(tmp);

                //Décodage de la commande -->ip:port:co command

               string[] cmd = data.Split(new char[] { ':' });
```

command → nb1 : signe : nb2

data sous fome de string essaie de les mettre dans un vecteur de string et fait un split

Sockets – TP2 –

cmd	Ip host	port	nb1	signe	nb2
	cmd[0]	cmd[1]	cmd[2]	cmd[3]	cmd[4]

```
            //adresse ip
            string host = cmd[0];

            //port de connexion
            int port = Int32.Parse(cmd[1]);

            //récupération de nb1 du signe et de nb2

            int nb1 = Int32.Parse(cmd[2]);

            string signe = cmd[3];

            int nb2 = Int32.Parse(cmd[4]);

            int resultat = 0;
         Console.WriteLine("Operation --> " + nb1 + " " + signe + " " + nb2);
            //execution de la commande
            switch (signe)
            {
                case "+": resultat = nb1 + nb2; break;
                case "-": resultat = nb1 - nb2; break;
                case "*": resultat = nb1 * nb2; break;
                case "/": resultat = nb1 / nb2; break;
                case "<>": resultat = 0; loop = false; break;
                default: resultat = 0; break;
            }//fin du switch
           string reponse = nb1 + " " + signe + " " + nb2 + " = " + resultat;

            //conversion du resultat en bytes
      byte[] rep =System.Text.Encoding.ASCII.GetBytes(reponse.ToCharArray());
            serveur.Send(rep, rep.Length, host, port);

        }//fin du while/
        //arret du serveur
        Console.WriteLine("Arret du serveur...");
        Console.WriteLine("Arret du serveur... *Entree* pour terminer.");
        Console.ReadLine();
    } } }
```

3. Exécution des projets :

1. Démarrer d'abord l'exécution du projet serveur : **UdpServeur** à partir de l'icône de débogage.
2. Démarrer ensuite l'exécution du projet client : **Client** à partir du dossier où se trouve le projet **Client**. On retrouve par défaut l'exécutable du projet dans le chemin : **C:\Users\User\Documents\Visual Studio 2012\Projects\TP2---Socket\Client\bin\Debug**.
3. Tester les deux projets avec un jeu d'essai de données de votre choix

Sockets – TP2 –

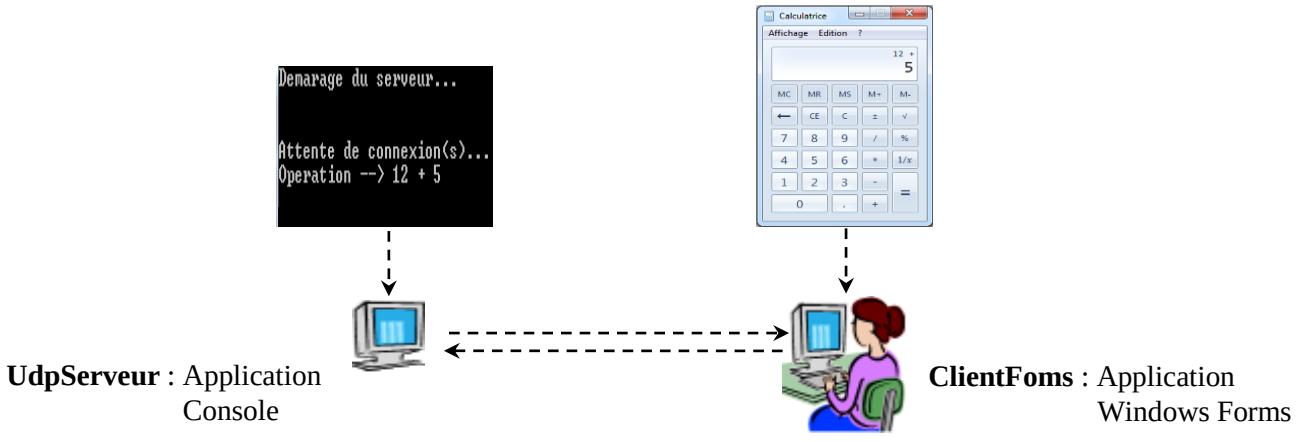
4. Redéployez les deux projets sur deux machines physiques, l'une hébergera le projet **UdpServeur** et l'autre le projet **Client**.
5. Créez pour cela un point d'accès avec votre **4G** que les deux machines partageront en commun.
6. Modifiez les adresses **IPs** (**Wifi**) dans les classes du projet **Client** seulement. **UdpServeur** reste inchangé.
7. Les classes du Client à modifier sont :
   - La classe client au niveau de l'instruction suivante :

     ```
     string req = "127.0.0.1:" + this.clientPort + ":" + command;
     ```
     Modifiez le 127.0.0.1 par l'adresse **ip (wifi)** de la machine qui héberge le projet **Client**.
   - La classe UDPClient au niveau de l'instruction suivante :

     ```
     client proc = new client(8081, "127.0.0.1", 8080);
     ```
8. Modifiez le 127.0.0.1 par l'adresse **ip (wifi)** de la machine qui héberge le projet **UdpServeur**.
9. Faites un test de communication des deux machines au niveau de la CMD : en faisant un **ping** avec l'adresse **ip** du réseau mobile **wifi** pour s'accéder mutuellement l'une à l'autre.
10. On peut connaitre l'adresse **ip** du réseau mobile **wifi** et autre à travers la commande **ipconfig**, dans la **cmd**.
11. Si le **ping** ne fonctionne pas, désactiver les pare-feu et d'éventuels l'antivirus sur les deux machines.
12. Si vous n'avez pas deux machines physiques pour tester, vous pouvez tester sur une seule machine qui hébergera le projet **UdpServeur** et installez une machine virtuelle telle que Linux pour héberger le **Client**.
13. Testez cette nouvelle configuration sur deux machines (l'une physique et l'autre virtuelle).

II- Communication Client/Serveur par socket TCP (Travail à faire)

Etendre la solution précédente en concevant le client comme une application Windows Forms et en mettant en œuvre une interaction entre le client et le serveur par sockets TCP déployée sur **deux machines** physiques.



UdpServeur : Application Console

ClientFoms : Application Windows Forms