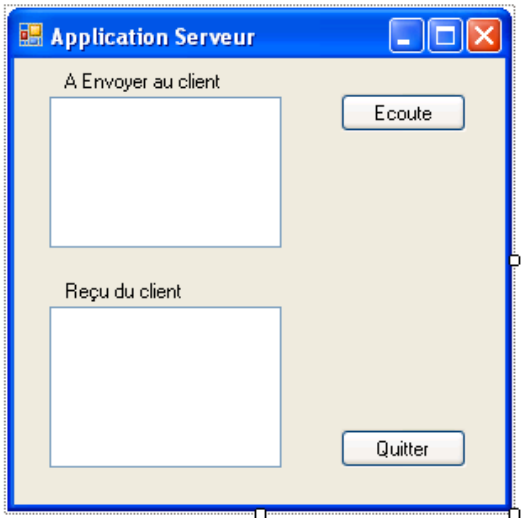
TP communication réseau par socket

I) Conception du serveur

1) Réalisation de l'interface graphique.



2) codage

- **Déclaration des directives "using" variables et constructeur de la classe du formulaire.**

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

namespace TestSocket
{
    public partial class Form1 : Form
    {

        Socket sock;
        Socket sockClient;



        public Form1()
        {
            InitializeComponent();
            //Création de l'objet socle de communication
            sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            //Association du socle à l'adresse IP de la machine et à un numéro de port.
            sock.Bind(new IPEndPoint(IPAddress.Parse("192.168.0.11"),4000));
        }
```

- **Gestionnaire d'événement click du bouton "Ecoute"**

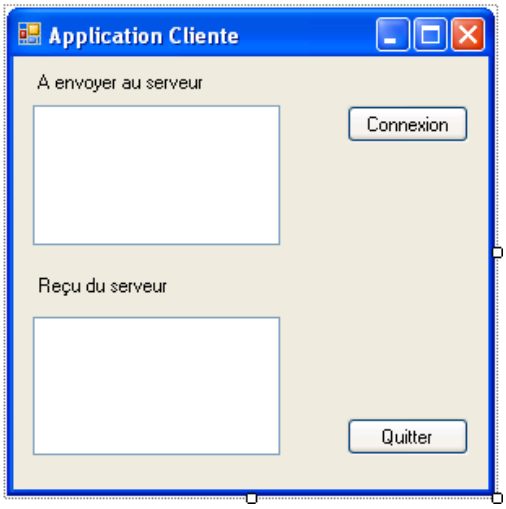
```
private void buttonConnecter_Click(object sender, EventArgs e)
{
    //Serveur à l'écoute
    sock.Listen(1);
    //Le serveur attend la demande de connexion du client (Accept bloquant)
    sockClient = sock.Accept();
    //le client est connecté, La variable sockClient permet de dialoguer avec le client
    //envoi d'un message au client
    string reponse = "Vous êtes connecté au serveur";
    byte[] tab = ASCIIEncoding.Default.GetBytes(reponse);
    sockClient.Send(tab);

}
```

- **Gestionnaire d'événement click du bouton "Quitter"**

```
private void buttonFermer_Click(object sender, EventArgs e)
{
    //Fin de l'émission et réception
    sockClient.Shutdown(SocketShutdown.Both);
    //Fermeture de la liaison par socle
    sockClient.Close();
    sock.Close();
    this.Close();
}
```

II) Conception du client.

1) Réalisation de l'interface graphique.



2) codage

- **Déclaration des directives "using" variables et constructeur de la classe du formulaire.**

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

namespace TestSocketClient
{

    public partial class FormClient : Form
    {
        Socket sock;


        public FormClient()
        {
            InitializeComponent();
            //création de l'objet socle de communication
            sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        }
```

- **Gestionnaire d'événement click du bouton "Connexion"**

```
private void buttonConnexion_Click(object sender, EventArgs e)
{
    //demande de connexion au serveur
    sock.Connect(new IPEndPoint(IPAddress.Parse("192.168.0.11"),4000));
    //Lecture de la réponse du serveur, création d'un tableau dynamique.
    byte[] buffer = new byte[50];
    //receive lit les données dans une mémoire tampon de réception
    //si le nombre de caractères envoyés par le serveur est > à 50, il faut
    //appeler de nouveau Receive().Receive()est bloquant (lecture synchrone)
    int n = sock.Receive(buffer);
    string recu = ASCIIEncoding.Default.GetString(buffer);
    textBoxReception.Text = recu;


}
```

- **Gestionnaire d'événement click du bouton "Quitter"**

```
private void buttonFermer_Click(object sender, EventArgs e)
{
    //fin de l'émission et reception
    sock.Shutdown(SocketShutdown.Both);
    //fermeture la liaison par socle
    sock.Close();
    //ferme l'application
    this.Close();

}
```

III) Améliorations

- **Modification de l'application serveur:**

L'application serveur affiche le nom de l'ordinateur et récupère automatiquement l'adresse IP de sa machine, et l' affiche.

L'application serveur peut dialoguer avec le client (émission, réception)

On peut renseigner dans l'interface graphique, le n° de port.

- **Modification de l'application cliente:**

L'application cliente utilise le nom du serveur à la place de l'adresse IP.

On peut renseigner dans l'interface graphique le nom du serveur et le port

L'application cliente peut dialoguer avec le serveur (émission, réception).

IV) Synthèse

- Réaliser une application serveur pouvant gérer plusieurs clients.
  On utilisera un thread pour chaque communication.