Hướng dẫn Lab 6

- **Server**

```
class ServerProgram
    {
        private IPAddress serverIP;
        public IPAddress ServerIP
        {
            get
            {
                return serverIP;
            }
            set
            {
                this.serverIP = value;
            }
        }
        private int port;
        public int Port
        {
            get
            {
                return this.port;
            }
            set
            {
                this.port = value;
            }
        }
        //delegate để set dữ liệu cho các Control
        //Tại thời điểm này ta chưa biết dữ liệu sẽ được hiển thị vào đâu nên
phải dùng delegate
        public delegate void SetDataControl(string Data);
        public SetDataControl SetDataFunction = null;
        public delegate void SetStatusControl(string Data);
        public SetStatusControl SetStatusFunction = null;
        Socket serverSocket = null;
        IPEndPoint serverEP = null;
        Socket clientSocket = null;
        //buffer để nhận và gởi dữ liệu
        byte[] buff = new byte[1024];
        //Số byte thực sự nhận được
        int byteReceive = 0;
        string stringReceive = "";
        public ServerProgram(IPAddress ServerIP, int Port)
        {
            this.ServerIP = ServerIP;
            this.Port = Port;
        }
        //Lắng nghe kết nối
        public void Listen()
        {
            serverSocket = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);
            serverEP = new IPEndPoint(ServerIP, Port);
            //Kết hợp Server Socket với Local Endpoint
            serverSocket.Bind(serverEP);
            //Lắng nghe kết nối trên Server Socket
            //-1: không giới hạn số lượng client kết nối đến
            serverSocket.Listen(-1);
            SetStatusFunction("Dang cho ket noi");
```

```
            //Bắt đầu chấp nhận Client kết nối đến
            serverSocket.BeginAccept(new AsyncCallback(AcceptScoket),
serverSocket);
        }
        //Hàm callback chấp nhận Client kết nối
        private void AcceptScoket(IAsyncResult ia)
        {
            Socket s = (Socket)ia.AsyncState;
            //Hàm Accept sẽ block server lại và chờ Client kết nối đến
            //Sau khi Client kết nối đến sẽ trả về socket chứa thông tin của
Client
            clientSocket = s.EndAccept(ia);
            string hello = "Hello Client";
            buff = Encoding.ASCII.GetBytes(hello);
            SetStatusFunction("Client " + clientSocket.RemoteEndPoint.ToString()
            + "da ket noi den");
            clientSocket.BeginSend(buff, 0, buff.Length, SocketFlags.None, new
            AsyncCallback(SendData), clientSocket);
        }
        private void SendData(IAsyncResult ia)
        {
            Socket s = (Socket)ia.AsyncState;
            s.EndSend(ia);
            //khởi tạo lại buffer để nhận dữ liệu
            buff = new byte[1024];
            //Bắt đầu nhận dữ liệu
            s.BeginReceive(buff, 0, buff.Length, SocketFlags.None, new
AsyncCallback(ReceiveData), s);
        }
        public void Close()
        {
            clientSocket.Close();
            serverSocket.Close();
        }
        private void ReceiveData(IAsyncResult ia)
        {
            Socket s = (Socket)ia.AsyncState;
            try
            {
                //Hàm EmdReceive sẽ bị block cho đến khi có dữ liệu trong TCP
buffer
                byteReceive = s.EndReceive(ia);
            }
            catch
            {
                //Trường hợp lỗi xảy ra khi Client ngắt kết nối
                this.Close();
                SetStatusFunction("Client ngat ket noi");
                this.Listen();
                return;
            }
            //Nếu Client shutdown thì hàm EndReceive sẽ trả về 0
            if (byteReceive == 0)
            {
                Close();
                SetStatusFunction("Clien dong ket noi");
            }
            else
            {
                stringReceive = Encoding.ASCII.GetString(buff, 0, byteReceive);
                SetDataFunction(stringReceive);
                //Sau khi Server nhận dữ liệu xong thì bắt đầu gởi dữ liệu xuống
cho Client
```

```
                s.BeginSend(buff, 0, buff.Length, SocketFlags.None, new
AsyncCallback(SendData), s);
            }
        }
    }
```

- **Client**

```
class ClientProgram
    {
        //delegate để set dữ liệu cho các Control
        //Tại thời điểm này ta chưa biết dữ liệu sẽ được hiển thị vào đâu nên
phải dùng delegate
        public delegate void SetDataControl(string Data);
        public SetDataControl SetDataFunction = null;
        //buffer để nhận và gởi dữ liệu
        byte[] buff = new byte[1024];
        //Số byte thực sự nhận được
        int byteReceive = 0;
        //Chuỗi nhận được
        string stringReceive = "";
        Socket serverSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint serverEP = null;
        //Lắng nghe kết nối
        public void Connect(IPAddress serverIP, int Port)
        {
            serverEP = new IPEndPoint(serverIP, Port);
            //Việc kết nối có thể mất nhiều thời gian nên phải thực hiện bất đồng
bộ
            serverSocket.BeginConnect(serverEP, new
AsyncCallback(ConnectCallback), serverSocket);
        }
        //Hàm callback chấp nhận Client kết nối
        private void ConnectCallback(IAsyncResult ia)
        {
            //Lấy Socket đang thực hiện việc kết nối bất đồng bộ
            Socket s = (Socket)ia.AsyncState;
            try
            {
                //Set dữ liệu cho Control
                SetDataFunction("Đang chờ kết nối");
                //Hàm EndConnect sẽ bị block cho đến khi kết nối thành công
                s.EndConnect(ia);
                SetDataFunction("Kết nối thành công");
            }
            catch
            {
                SetDataFunction("Kết nối thất bại");
                return;
            }
            //Ngay sau khi kết nối xong bắt đầu nhận câu chào từ Server gởi xuống
            s.BeginReceive(buff, 0, buff.Length, SocketFlags.None, new
AsyncCallback(ReceiveData), s);
        }
        private void ReceiveData(IAsyncResult ia)
        {
            Socket s = (Socket)ia.AsyncState;
            byteReceive = s.EndReceive(ia);
            stringReceive = Encoding.ASCII.GetString(buff, 0, byteReceive);
            SetDataFunction(stringReceive);
        }
```

```
    private void SendData(IAsyncResult ia)
    {
        Socket s = (Socket)ia.AsyncState;
        s.EndSend(ia);
        //khởi tạo lại buffer để nhận dữ liệu
        buff = new byte[1024];
        //Bắt đầu nhận dữ liệu
        s.BeginReceive(buff, 0, buff.Length, SocketFlags.None, new
        AsyncCallback(ReceiveData), s);
    }
    //Hàm ngắt kết nối
    public bool Disconnect()
    {
        try
        {
            //Shutdown Soket đang kết nối đến Server
            serverSocket.Shutdown(SocketShutdown.Both);
            serverSocket.Close();
            return true;
        }
        catch
        {
            return false;
        }
    }
    //Hàm gởi dữ liệu
    public void SendData(string Data)
    {
        buff = Encoding.ASCII.GetBytes(Data);
        serverSocket.BeginSend(buff, 0, buff.Length, SocketFlags.None, new
        AsyncCallback(SendToServer), serverSocket);
    }
    //Hàm CallBack gởi dữ liệu
    private void SendToServer(IAsyncResult ia)
    {
        Socket s = (Socket)ia.AsyncState;
        s.EndSend(ia);
        buff = new byte[1024];
        s.BeginReceive(buff, 0, buff.Length, SocketFlags.None, new
        AsyncCallback(ReceiveData), s);
    }
}
```