برنامه نویسی سوکت
Socket Programming

مدرس:
مهندس جواد فرقانی
ایمیل:
jforghani@live.com

برنامه شبکه

- هر برنامه شبکه از دو بخش تشکیل شده
  - سرور (خدمات دهنده)
    - آدرس مشخصی دارد
    - منتظر درخواست است
  - مشتری (خواهان خدمات)
    - آدرس مشخص ندارد
    - درخواست خود را به سرور می دهد
  - سوکت برکلی یک روش استاندارد برای برنامه شبکه

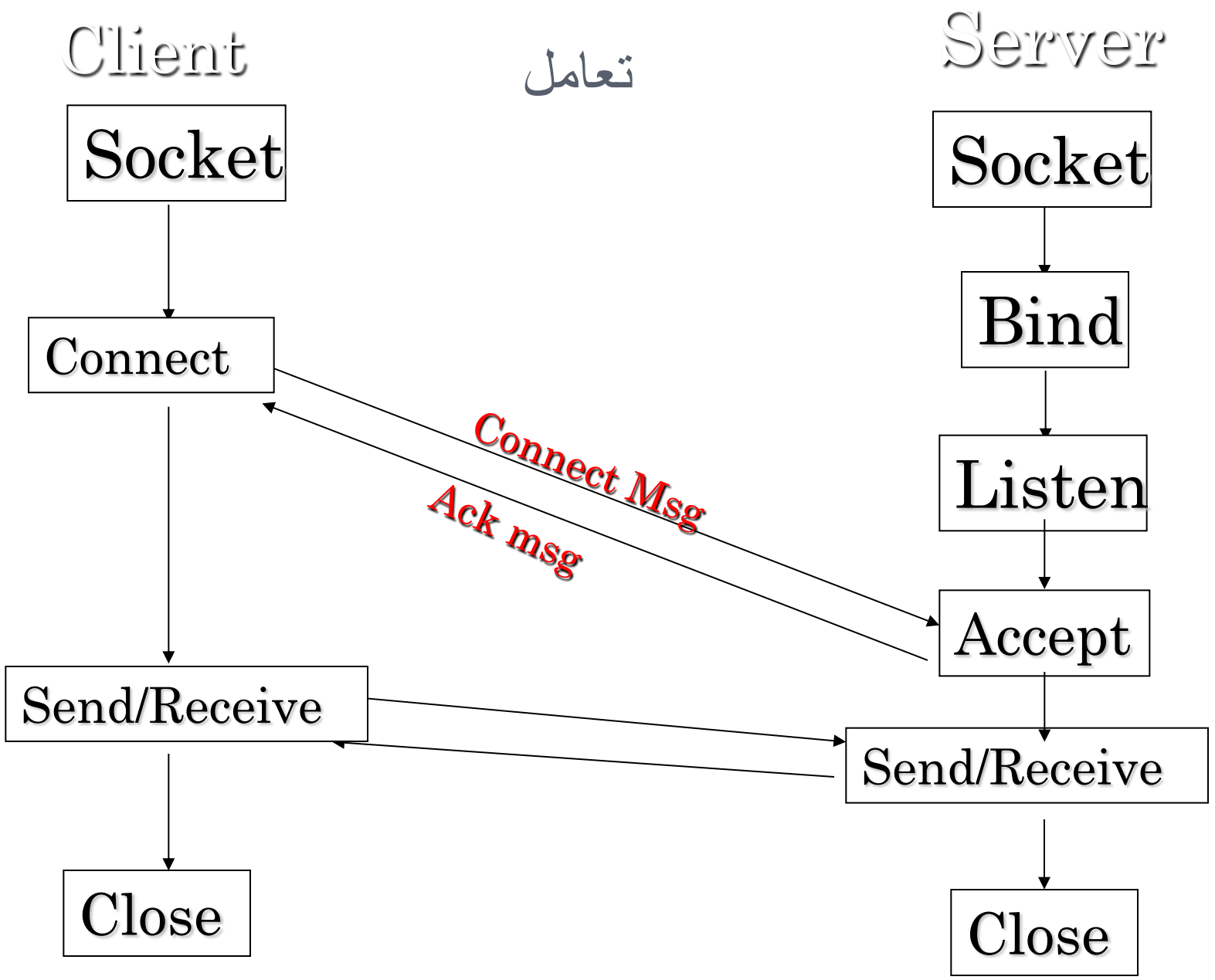
گامهای برنامه شبکه (سرور TCP)

- ایجاد سوکت
- نسبت دادن آدرس
- انتظار برای تقاضا و مدیریت تقاضا
- پذیرش تقاضا
  - انجام تقاضا
  - ارسال و دریافت
  - پایان تقاضا
- پایان انتظار و صف کردن تقاضا

گامهای برنامه شبکه (مشتری TCP)

- ایجاد سوکت
- ارسال تقاضا به سرور
  - انجام تقاضا
- پایان تقاضا

مثال از برنامه شبکه


Client
تعامل
Server
Socket
Socket
Connect
Bind
Connect Msg
Ack msg
Listen
Accept
Send/Receive
Send/Receive
Close
Close

مفاهیم

- سوکت
  - مفهومی انتزاعی برای دسترسی به امکانات لایه انتقال
- آدرس آی پی
  - مشخص کننده آدرس کامپیوتری که سرور روی آن قرار دارد
- شماره پورت
  - مشخص کننده برنامه ای که روی سرور اجرا می شود

نوع داده های مورد نیاز

- سوکت گوش دهنده
  - System.Net.Sockets.TcpListener
  - برای گوش دادن به درخواستها
- سوکت معمولی (کلاس سوکت در این فضای نام قرار دارد)
  - System.Net.Sockets.Socket
- سوکت مشتری
  - System.Net.Sockets.TcpClient

- Socket(AddressFamily *af, SocketType st,ProtocolType pt)*

- *AddressFamily*: نوع شبکه را مشخص می کند
  - برای *IP* نرمال باید از مقدار AddressFamily.InterNetwork استفاده کنیم.

- *SocketType*: نوع ارتباط داده ها را مشخص می کند
  - UDP Dgram
  - *TCP Stream*
  - *ICMP Raw*
- *ProtocolType*: نوع پروتکل شبکه را مشخص می کند

برنامه سمت کاربر

```
byte[] data = new byte [2048];
string input, stringData;
IPEndPoint ipep = new IPEndPoint (IPAddress.Parse("127.0.0.1"), 5060);
Socket server = new Socket(AddressFamily.InterNetwork,
    SocketType.Stream,ProtocolType.Tcp);

server.Connect(ipep);

int recv = server.Receive(data);
stringData = Encoding.ASCII.GetString(data, 0, recv);

textBox1.Text = stringData;
```

برنامه سمت سرور

```
IPAddress ip;
IPEndPoint ie;
byte[] data = new byte[2048];
ip = IPAddress.Parse("127.0.0.1");
ie=new IPEndPoint(ip,5060);

Socket s=new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

s.Bind(ie);
s.Listen(6);

Socket client = s.Accept();
IPEndPoint clientEp = (IPEndPoint)client.RemoteEndPoint;

string welcome = "Welcome to my test server";
data = Encoding.ASCII.GetBytes(welcome);
client.Send(data, data.Length, SocketFlags.None);
```

جریان و تبادل داده ها

- جریان
  - System.Net.Sockets.NetworkStream
- StreamReader
  - برای خواندن از جریان
- StreamWriter
  - برای نوشتن در جریان

توابع

- ایجاد System.Net.Sockets.TcpListener: بر اساس شماره پورت

```
TcpListener listener = new TcpListener(local_port);
```

- شروع انتظار برای اتصال

```
listener.Start();
```

گامهای سمت سرور

- دریافت تقاضا
- Socket soc = listener.AcceptSocket();
- ایجاد جریان برای انجام تقاضا
- Stream s = new NetworkStream(soc);
- بستن جریان
- s.Close();
- بستن سوکت
- soc.Close();

گامهای سمت مشتری

- ایجاد System.Net.Sockets.TcpClient براساس شماره پورت و آدرس آی پی سرور

```
TcpClient client = new TcpClient(host, port);
```

- تقاضای اتصال

```
Client. Connect("127.0.0.1", 2030);
```

- ایجاد جریان برای انجام تقاضا

```
Stream s = client.GetStream()
```

- بستن جریان

```
s.Close();
```

- بستن سوکت مشتری

```
client.Close();
```

توابع مربوط به انجام تقاضا

- ابتدا یک جریان از اتصال شبکه می گیریم:
  - Stream s = new NetworkStream(soc);
- سپس جریانهای نوشتن و خواندن آنرا مشخص می کنیم:
  - StreamReader sr = new StreamReader(s);
  - StreamWriter sw = new StreamWriter(s);

- برای ارسال کافیست:

```
sw.WriteLine("رشته");
sw.Flush();
```

- برای دریافت کافیست:

```
sr.ReadLine();
```

برنامه سرور

```
using System;
using System.Threading;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Configuration;

class EmployeeTCPServer{
    static TcpListener listener;
    const int LIMIT = 5; //5 concurrent clients
```

برنامه سرور

```
public static void Main(){
    listener = new TcpListener(2055);
    listener.Start();

      for(int i = 0;i < LIMIT;i++){
        Thread t = new Thread(new ThreadStart(Service));
        t.Start();
    }
}

public static void Service(){
    while(true){
        Socket soc = listener.AcceptSocket();
           try{
            Stream s = new NetworkStream(soc);
            StreamReader sr = new StreamReader(s);
            StreamWriter sw = new StreamWriter(s);
            sw.AutoFlush = true; // enable automatic flushing
            sw.WriteLine("{0} Employees available",
                  ConfigurationSettings.AppSettings.Count);
```

برنامه سرور

```
while(true){
                    string name = sr.ReadLine();
                    if(name == "" || name == null) break;
                    string job =
                        ConfigurationSettings.AppSettings[name];
                    if(job == null) job = "No such employee";
                    sw.WriteLine(job);
                }
                s.Close();
            }catch(Exception e){
                                    }
                        soc.Close();
        }
    }
}
```

برنامه مشتری

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

class EmployeeTCPClient{
public static void Main(string[] args)
{
TcpClient client = new TcpClient(args[0],2055);
try{ Stream s = client.GetStream();
StreamReader sr = new StreamReader(s);
StreamWriter sw = new StreamWriter(s);
sw.AutoFlush = true; Console.WriteLine(sr.ReadLine());
while(true){
Console.Write("Name: ");
string name = Console.ReadLine();
 sw.WriteLine(name);
if(name == "") break;
Console.WriteLine(sr.ReadLine()); }
 s.Close();
}finally{ client.Close(); } } }
```

گامهای برنامه سرور در UDP

- ایجاد سوکت
- نسبت دادن آدرس
  - انجام تقاضا
  - ارسال و دریافت
  - پایان تقاضا
- پایان کار سوکت

گامهای برنامه مشتری در UDP

- ایجاد سوکت
- انجام تقاضا
  - ارسال و دریافت
- پایان کار سوکت

- برنامه نویس باید دو کار برای برنامه سمت سرور انجام دهد:
  ۱. ساخت یک شی از نوع سوکت
  ۲. مقید کردن سوکت به یک IPEndPoint

- بعد از این شما می توانید به ارسال و دریافت داده بپردازید، اما شما برای ارسال و دریافت نمی توانید از متد های Receive() و Send() استفاده کنید بلکه باید از دو متد جدید ReceiveFrom() و SendTo() استفاده کنیم.

- شکل کلی متد بصورت زیر است:

- SendTo(byte[] *data,int Offset,int Size,SocketFlags,* Flags,EndPoint *Remote)*

- ReceiveFrom(byte[] data, ref Endpoint Remote)

Server program

```
int recv;
byte[] data = new byte[1024];
IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);
Socket newsock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
newsock.Bind(ipep);

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint Remote = (EndPoint)(sender);
recv = newsock.ReceiveFrom(data, ref Remote);
Console.WriteLine("Message received from {0}:",
Remote.ToString());
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
string welcome = "Welcome to my test server";
data = Encoding.ASCII.GetBytes(welcome);
```

```
newsock.SendTo(data, data.Length, SocketFlags.None, Remote);
while (true)
{
    data = new byte[1024];
    recv = newsock.ReceiveFrom(data, ref Remote);
    Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
    newsock.SendTo(data, recv, SocketFlags.None, Remote);
}
```

Client Program

```
byte[] data = new byte[1024];
string input, stringData;
IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
Socket server = new Socket ( AddressFamily.InterNetwork,
    SocketType.Dgram,
ProtocolType.Udp);
string welcome = "Hello, are you there?";
data = Encoding.ASCII.GetBytes(welcome);
server.SendTo(data, data.Length, SocketFlags.None, ipep);
IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
EndPoint Remote = (EndPoint)sender;
data = new byte[1024];
int recv = server.ReceiveFrom(data, ref  Remote);
Console.WriteLine("Message received from {0}:",
Remote.ToString());
Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
```

```
while (true)
{
    input = Console.ReadLine();
    if (input == "exit")
    break;
    server.SendTo(Encoding.ASCII.GetBytes(input), Remote);
    data = new byte[1024];
    recv = server.ReceiveFrom(data, ref Remote);
    stringData = Encoding.ASCII.GetString(data, 0, recv);
    Console.WriteLine(stringData);
}
Console.WriteLine("Stopping client");
server.Close();
```

Broad Cast

- در واقع زمانی که بخواهید یک پیغام را برای همه ی کامپیوتر های شبکه تا یک محدوده مجاز ارسال کنید به این کار Broad Cast می گویند.

- برای ارسال Broad Cast نمی توان از پروتکل TCP استفاده کرد زیرا در TCP ارتباط دو دستگاه باید خصوصی باشد، بدین ترتیب از UDP استفاده می گردد.

- انواع Broad Cast :
  - Local Broad Cast
  - Global Broad Casr

Client 1

```
Socket  sock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram,ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Broadcast, 9050);
byte[] data = Encoding.ASCII.GetBytes("This is a test message");
sock.SendTo(data, iep);
sock.Close();
```

Client 2

```
Socket  sock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram,ProtocolType.Udp);
IPEndPoint iep1 = new IPEndPoint(IPAddress.Broadcast, 9050);
IPEndPoint iep2 = new IPEndPoint(IPAddress.Parse("192.168.0.255"),
    9050);
byte[] data = Encoding.ASCII.GetBytes("This is a test message");
sock.SetSocketOption(SocketOptionLevel.Socket,SocketOptionName.
    Broadcast, 1);
sock.SendTo(data, iep1);
sock.SendTo(data, iep2);
sock.Close();
```

Server

```
Socket sock = new Socket(AddressFamily.InterNetwork,
    SocketType.Dgram, ProtocolType.Udp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
sock.Bind(iep);
EndPoint ep = (EndPoint)iep;
byte[] data = new byte[1024];
int recv = sock.ReceiveFrom(data, ref ep);
string stringData = Encoding.ASCII.GetString(data, 0, recv);
data = new byte[1024];
recv = sock.ReceiveFrom(data, ref ep);
stringData = Encoding.ASCII.GetString(data, 0, recv);
sock.Close();
```