Intro to C# Network Programming

Overview of Steps

Server

Client

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination

Client

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination

Client

1. Start a client that establishes a connection to the server
   - Connects to an ip address and port combination

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination
2. Accept and establish connection with client

Client

1. Start a client that establishes a connection to the server
   - Connects to an ip address and port combination

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination
2. Accept and establish connection with client

Client

1. Start a client that establishes a connection to the server
   - Connects to an ip address and port combination
2. Send a message to the server

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination
2. Accept and establish connection with client
3. Receive and process message from a client

Client

1. Start a client that establishes a connection to the server
   - Connects to an ip address and port combination
2. Send a message to the server

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination
2. Accept and establish connection with client
3. Receive and process message from a client
4. Send response to the client

Client

1. Start a client that establishes a connection to the server
   - Connects to an ip address and port combination
2. Send a message to the server

Overview of Steps

Server

1. Start a server that establishes a listener for incoming client connections
   - Listens on an ip address and port combination
2. Accept and establish connection with client
3. Receive and process message from a client
4. Send response to the client

Client

1. Start a client that establishes a connection to the server
   - Connects to an ip address and port combination
2. Send a message to the server
3. Receive a message from the server

Server – Create Network Listener

```
// Binding to 'localhost'
IPHostEntry ipHost = Dns.GetHostEntry(Dns.GetHostName());
IPAddress ipAddr = ipHost.AddressList[0];
IPEndPoint localEndPoint = new IPEndPoint(ipAddr, port);

// Create a stream oriented connection (TCP)
Socket listener = new Socket(
    ipAddr.AddressFamily,
    SocketType.Stream,
    ProtocolType.Tcp);
listener.Bind(localEndPoint);

// allow up to 10 pending connections
listener.Listen(10);
```

Server – Wait for Client Connections

```
HashSet<Socket> clients = new HashSet<Socket>();


// This is a blocking call to wait for a connection
Socket client = listener.Accept();

// Add this socket to a container for future use
clients.Add(client);
```

Server – Wait for Client Message

```
// These are defined elsewhere and muthor of assumptions is that
// when a client connection is established, a blank string entry
// is made for that client.
HashSet<Socket> clients = new HashSet<Socket>();
Dictionary<Socket, string> messages = new Dictionary<Socket, string>();

foreach (var client in clients)
{
    if (client.Available > 0)
    {
        byte[] buffer = new byte[client.Available];
        client.Receive(buffer);
        clientMessages[client] += Encoding.ASCII.GetString(buffeer);
    }
}
```

Server – Send a Message to The Client

```
string message = "Here is some important info"
byte[] bytes = Encoding.ASCII.GetBytes(message);

client.Send(bytes);
```

Client – Establish Connection to a Server

```
// Setup the server IP address & Port combination
// See sample code for this function
IPAddress address = parseIPAddress("localhost");
IPEndPoint endpoint = new IPEndPoint(address, 3001);

Socket server = new Socket(
    endPoint.AddressFamily,
    SocketType.Stream,
    ProtocolType.Tcp);
server.Connect(remoteEndPoint);
```

Client – Send a Message to The Server

```
string message = "Network programming is fun!"
byte[] bytes = Encoding.ASCII.GetBytes(message);

server.Send(bytes);
```

Client – Receive a Message from The Server

```
if (server.Available > 0)
{
    byte[] bytes = new byte[server.Available];
    int bytesReceived = server.Receive(bytes);
    string message = Encoding.ASCII.GetString(bytes, 0, bytesReceived);
}
```

Other Things

- Need to know when a full message has been received
  - First two bytes can indicate the total bytes in a message
  - Place a marker at the end of a message (demonstrated in the code sample)
- Race-Conditions (multi-threading and async)
  - Can’t change a container while iterating through it
  - Multi-Threading & Async operations must be thread safe
- Typically don’t want to do IO operations on the main thread
  - IO is slow (relatively speaking)