**Socket Communication Overview**(excerpted from <https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm>)

Your computer is like an apartment building:



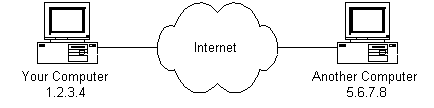
it has a physical "street" address (the internet protocol (ip) address - IPCONFIG.EXE)

and it has apartments, which are like the communication ports

it has 65,535 Transmission Control Protocol (TCP) ports and 65,535 User Datagram Protocol (UDP) ports

* TCP maintains a direct connection between sender and receiver (lossless transmission)
* UDP does not maintain a direct connection (throws the message over the wall and hopes for the best!) The advantage is very little overhead.

Each apartment is like a port and serves a different communication function. Many are "hard-coded" and standard across all computers. Port 80 is always used for HTTP (web-based) communication. Port 21 is always used for FTP (file transfer protocol) communication. There are also POP and SMTP ports for email. This is why you can have multiple communication applications running simultaneously (browser, email, FileZilla, MineCraft). Therefore the resident in apartment 995 is solely tasked with handling your email. Personally I'd rather live in apartment 80!



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Protocol Layer** | **Comments** | | Application Protocols Layer | Protocols specific to applications such as WWW, e-mail, FTP, etc. | | Transmission Control Protocol Layer | TCP directs packets to a specific application on a computer using a port number. | | Internet Protocol Layer | IP directs packets to a specific computer using an IP address. | | Hardware Layer | Converts binary packet data to network signals and back. (E.g. ethernet network card, modem for phone lines, etc.) |      |  | | --- | | Diagram 2 | |  |   **The Data Packet is Built from Right to Left**  Diagram 9  **Networking Infrastructure (TRACERT.EXE)**   |  | | --- | | Diagram 3 |  Internet Service Provider (ISP)Internet Infrastructure The Internet backbone is made up of many large networks which interconnect with each other. These large networks are known as **Network Service Providers** or **NSP**s. Some of the large NSPs are UUNet, CerfNet, IBM, BBN Planet, SprintNet, PSINet, as well as others. These networks **peer** with each other to exchange packet traffic. Each NSP is required to connect to three **Network Access Points** or **NAP**s. At the NAPs, packet traffic may jump from one NSP's backbone to another NSP's backbone. NSPs also interconnect at **Metropolitan Area Exchanges** or **MAE**s. MAEs serve the same purpose as the NAPs but are privately owned. NAPs were the original Internet interconnect points. Both NAPs and MAEs are referred to as Internet Exchange Points or **IX**s. NSPs also sell bandwidth to smaller networks, such as ISPs and smaller bandwidth providers. Below is a picture showing this hierarchical infrastructure.   |  | | --- | | Diagram 4 |  The Internet Routing Hierarchy The information used to get packets to their destinations are contained in routing tables kept by each router connected to the Internet.  **Routers are packet switches.** A router is usually connected between networks to route packets between them. Each router knows about it's sub-networks and which IP addresses they use. The router usually doesn't know what IP addresses are 'above' it. The black boxes connecting the backbones are routers. The larger NSP backbones at the top are connected at a NAP. Under them are several sub-networks, and under them, more sub-networks. At the bottom are two local area networks with computers attached.   |  | | --- | | Diagram 5 | |  |  Domain Names and Address Resolution The **Domain Name Service** or **DNS** is a distributed database which keeps track of computer's names and their corresponding IP addresses on the Internet.  Many computers connected to the Internet host part of the DNS database and the software that allows others to access it. These computers are known as DNS servers. No DNS server contains the entire database; they only contain a subset of it. If a DNS server does not contain the domain name requested by another computer, the DNS server re-directs the requesting computer to another DNS server.   |  | | --- | | Diagram 6 |   **TCP Header**   |  | | --- | | Diagram 7 | |  |   Notice that there is no place for an IP address in the TCP header. This is because TCP doesn't know anything about IP addresses. TCP's job is to get application level data from application to application reliably. The task of getting data from computer to computer is the job of IP.    **Internet Protocol**  Unlike TCP, **IP is an unreliable, connectionless protocol**. IP doesn't care whether a packet gets to it's destination or not. Nor does IP know about connections and port numbers. **IP's job is too send and route packets to other computers**. IP packets are independent entities and may arrive out of order or not at all. It is TCP's job to make sure packets arrive and are in the correct order. About the only thing IP has in common with TCP is the way it receives data and adds it's own IP header information to the TCP data. The IP header looks like this:   |  | | --- | | Diagram 8 | | Diagram 8 |   Above we see the IP addresses of the sending and receiving computers in the IP header. Below is what a packet looks like after passing through the application layer, TCP layer, and IP layer. The application layer data is segmented in the TCP layer, the TCP header is added, the packet continues to the IP layer, the IP header is added, and then the packet is transmitted across the Internet.   |  | | --- | | Diagram 9 | |  | |

**TCP Socket Methods**

Creating a Socket

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

// if just attaching to a local port, then can use any address

IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Any, myPort);

// if we need to attach to a specific ip address

IPAddress iPAddress = IPAddress.Parse(targetIp);

where targetIp is a string like "192.168.1.3"

IPEndPoint localEndPoint = new IPEndPoint(iPAddress, targetPort);

// bind the socket to the IP address and Port number

listener.Bind(localEndPoint);

// start the socket to listen for messages from up to 300 clients

listener.Listen(300);

We used the StreamReader class and its ReadLine method in the MadLibs assignment to read the MadLibs template file:

StreamReader input;

// open the template file to count how many Mad Libs it contains

input = new StreamReader("e:\\templates\\MadLibsTemplate.txt");

string line = null;

// read 1 line at a time (a line ends with '\n')

while ((line = input.ReadLine()) != null)

{

++numLibs;

}

Sockets use StreamReader and StreamWriter to read and write data from and to Sockets:

Reading:

Socket client = listener.Accept();

Stream netStream = new NetworkStream(client);

StreamReader reader = new StreamReader(netStream);

string result = reader.ReadToEnd();

reader.Close();

netStream.Close();

client.Close();

Writing:

IPAddress iPAddress = IPAddress.Parse(targetIp);

IPEndPoint remoteEndPoint = new IPEndPoint(iPAddress, targetPort);

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

server.Connect(remoteEndPoint);

Stream netStream = new NetworkStream(server);

StreamWriter writer = new StreamWriter(netStream);

string msg = loginTextBox.Text + ": " + textBox.Text;

writer.Write(msg.ToCharArray(), 0, msg.Length);

writer.Close();

netStream.Close();

server.Close();