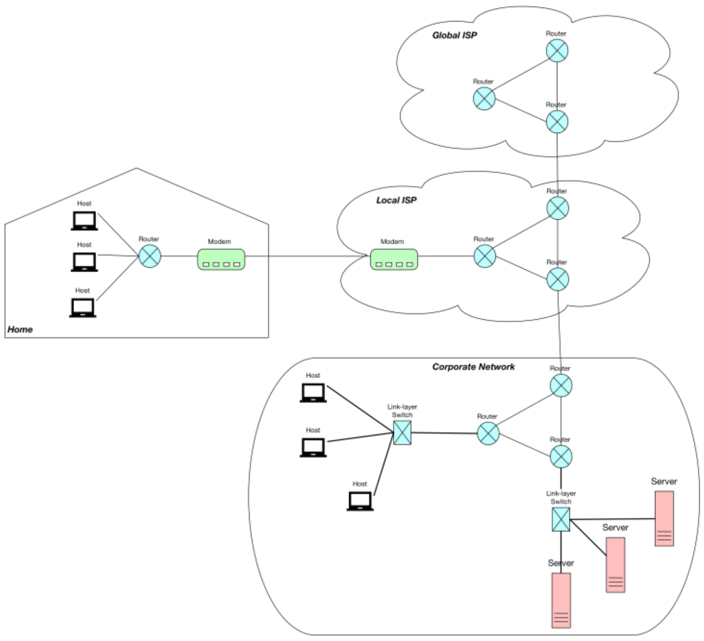
Networking

Introduction

THIS DOCUMENT COVERS

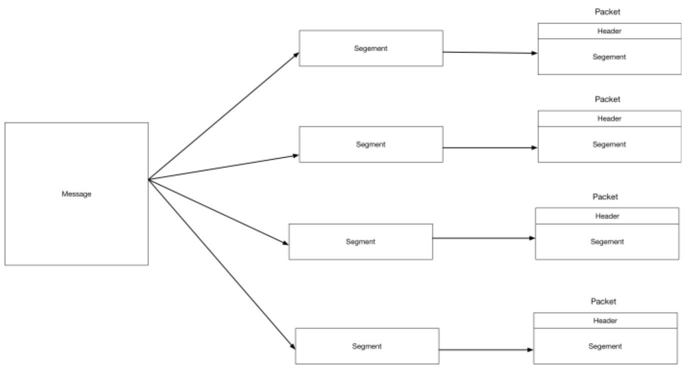
The Internet



A network consists of a set of computing devices known as end systems or hosts connected by communication links and packet switches. Hosts can be further classified as clients or servers.

Different communication links utilise different mediums such as coaxial cable, copper wire, optical fibre and radio waves. Each medium has its own transmission rate measured in bits per second.

An end system that wants to transmit a message first breaks it into smaller parts known as segments. A header is added to each segment to create a packet. The packets are then sent through the network and are reassembled into the original message at the destination.



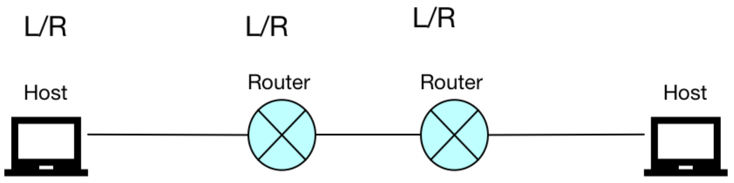
Packet switches such as routers and link level switches take packets from one of their incoming communications links and then route them on to one of their outgoing communication links.

Switching

In order to make it from source to destination packets pass through communication links and packet switches (routers). If a source or intermediate packet switch needs to send a packet of size L bits over a communication link with transmission rate of R bps the time to transmit the packet is L/R seconds. Because each packet switch must receive a whole packet before it can start to transmit the first bit of the packet onto the outbound communication link. If we are sending one packet along a route consisting of N identical links (N-1 routers) the total time taken is

$$N\frac{L}{R}$$

In the special case of three communication links and two routers it takes $3\frac{L}{R}$ for the packet to arrive at the destination (assuming no propagation delay)



If we have P packets then we need

Streams

A stream works at the level of bytes. Any data items need to be converted to bytes before they can be written to a stream.

```
var buffer = new byte[16];

using (Stream s = new MemoryStream(buffer))
{
        // We want to write two char to a stream. We first
        // need to use an encoding to convert the chars
        // to bytes
        char[] ab = {'a', 'b'};
        byte[] encodedBytes = Encoding.UTF8.GetBytes(ab);
        s.Write(encodedBytes, 0, encodedBytes.Length);

        // On the other side we use the encoding convert
        // the raw bytes back to characters.
        s.Position = 0;
        byte[] destBytes = new byte[256];
        s.Read(destBytes,0,encodedBytes.Length);
        char[] destChars = Encoding.UTF8.GetChars(destBytes,0,encodedBytes.Length);
}
```

Adapters can be used to read/write more complex types to the stream.

Transport level protocols

TCP

TCP is a connection-oriented transport level protocol which is used by HTTP amongst other application level protocols. When working at the transport level the developer must work out a protocol that specified at which times each side reads/writes.

```
var listener =
    new TcpListener(IPAddress.Parse("127.0.0.1"), 13000);

listener.Start();

            using TcpClient c = listener.AcceptTcpClient();
            using NetworkStream n = c.GetStream();
            using BinaryReader reader = new BinaryReader(n);
            using BinaryWriter writer = new BinaryWriter(n);

var msg = reader.ReadString();
Console.WriteLine(msg);
writer.Write(msg);
writer.Flush();
```

And the echo client is then

```
using TcpClient client = new TcpClient("localhost", 13000);
using NetworkStream n = client.GetStream();
BinaryWriter w = new BinaryWriter(n);
w.Write("Hello World");
w.Flush();
Console.WriteLine($"Client Received: {new
BinaryReader(n).ReadString()}");
```

It is more normal to work at a higher level than the transport layer. The following section details application level protocols.

Application level protocols

https://www.html5rocks.com/en/tutorials/websockets/basics/#toc-introduction-lowlatency

HTTP Long Polling

The client makes a normal request to a web server. The server then waits until it has a valid response before replying. As soon as the client gets the response it immediately sends another long polling request to the server. This system ensures a server always has a connection available to provide data to the client. Because there is a long lag between the client makes a long polling request and getting data back this approach is also known as a handing GET.

Web Sockets

Web Sockets use a TCP connection to provide a bi-directional communication channel between the client and server. Client and server must follow a specific handshake after which they can efficiently over a duplex channel.

The classic use case for using web sockets is chat application where a duplex connection is necessary.

Server Sent Events

Server sent events use a persistent long-term connection to enable real time data to be passed from the server to the client. Unlike web sockets the client cannot pass data to the server over a web socket. Server-Sent Events is a standard part of HTML5 via the Event Source API. SSE could be used for scenarios where we only need the server to send updates to the client such as a stock pricing application.

Rest