CS312 - Data Communications & Computer Networks

MACS, USP.

Hugh Anderson

February 10, 2000

Preface

In the study of any structured discipline, it is necessary to:

- appreciate the background to the discipline,
- know the terminology, and practice
- understand elements of the framework (hence the word *structured*).

Data communication is no different, so we begin by looking to the past, and studying the 'history' of data communication. Data communication is just one kind of communication, so we continue with general communication theory (Fourier, Nyquist and Shannon).

We also look at the equipment in current use before considering elements of data communications within a framework, known as the **OSI reference model**.

Contents

Chapter 1

Background

1.1 Prehistory

Data communication techniques predate computers by at least a hundred years - for example the MORSE code for communication over telegraph wires shown in Table 1.1. Long before we had radio, telegraph wires carried messages from one end of a country to another. At each end of the wire, the telegraph operators used MORSE to communicate.

MORSE of course is just an encoding technique. The basic elements of the code are two signals - one with a short duration, and one long. The signals were transmitted by just switching on and off an electric current (with a switch), and were received by checking the deflection of a magnet. Each letter is made from some sequence of short and long signals. If you examine the codes, you find that the most common letters are the ones with the shortest codes.

The most common letters in western languages are (in order): E T A O I N S H R D L U ... and sure enough - the E is the shortest code (a single .). T is next with -, followed by A (.-) and so on. Obviously when the MORSE code was developed, someone was concerned with efficiency.

Ham radio enthusiasts (Hams) use MORSE, but with higher level controls (protocols). They use short codes to encode commonly used requests or statements. They are known as the 'Q' codes

Letter	Code	Letter	Code	Letter	Code	Letter	Code
A	.-	B	-...	C	-.-.	D	-..
E	.	F	..-.	G	–.	H	....
I	..	J	.—	K	-.-	L	.-..
M	–	N	-.	O	—	P	.–.
Q	–.-	R	.-.	S	...	T	-
U	..-	V	...-	W	.–	X	-..-
Y	-.–	Z	–..				

Table 1.1: Morse Code.

Morse	Voice	Meaning
K	Go ahead.	Anyone go ahead
AR	Over.	Use at end of transmission before contact made
AS	Stand by.	Temporary interruption
R	Roger!	Transmission received OK
SK	Clear.	End of contact
CL	Closing down.	Shutting down transmitter

	Caller	Callee
Morse Voice	CQ CQ DE ZL2ASB K CQ this is ZL2ASB, go ahead.	
Morse Voice		ZL2ASB DE ZL2QW AR ZL2ASB from ZL2QW, over

Table 1.2: Ham calls.

and some are amusing. In table 1.2 are some of the common codes used by Hams. When a Ham wishes to make contact with anyone else, the call is as shown above.

- CQ is a request for contact
- ZL2ASB is the call sign
- K means anyone go ahead.
- AR means you want only one reply.

  In datacommunication terminology, Ham MORSE and voice transmission is asynchronous and half duplex. (See section 3)

The list of special *protocol* messages given above is by no means complete. It is also not particularly well structured or documented, and it is possible for Hams to *talk over the top of each other* without much difficulty. This of course does not normally matter. However simple protocols like this can cause problems.

In England in 1861, a poorly constructed protocol failed after 21 years of operation. In the ensuing mess, over 20 people died and a large number of people were hospitalized. The protocol's function was to ensure that only one train could be in the Clayton Tunnel (Ref Figure 1.1) at a time. At each end of the tunnel was a signal box, with a highly trained signaller, and some hi-tech (for the 19th century) signalling equipment involving a three-way switch and an indicator. The signaller could indicate any of three situations to the other signaller: (i) nothing at all, (ii) Train in Tunnel, and (iii) Tunnel Clear.

The signallers had a red flag which they would place by the track to signal to the train drivers to stop, and everyone followed the following protocol:

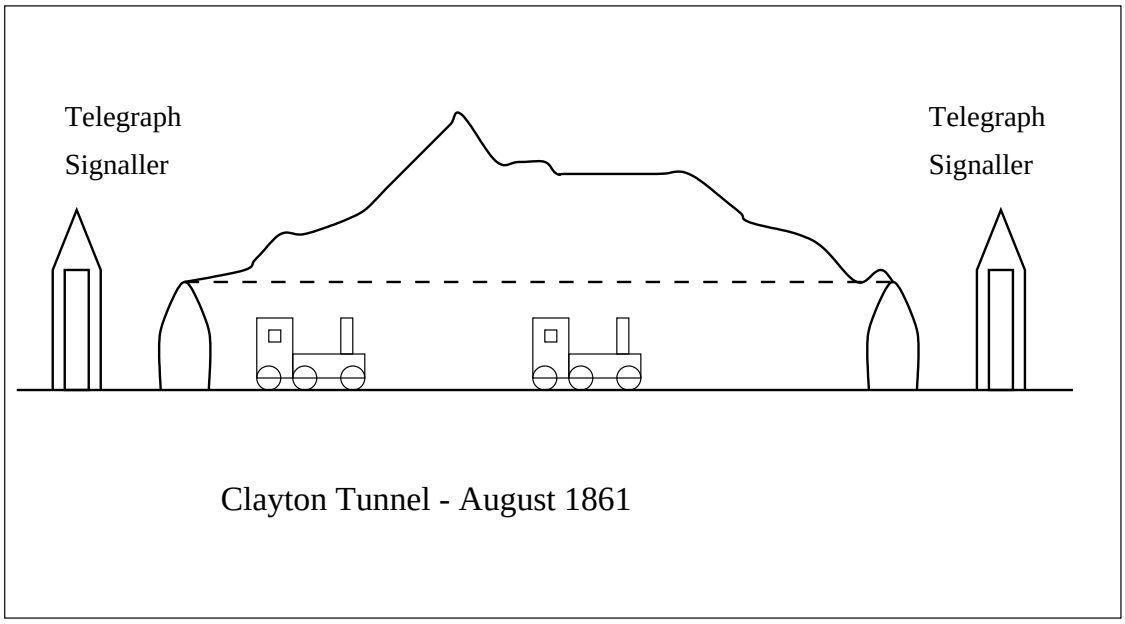


Figure 1.1: Recipe for disaster - the Clayton Tunnel protocol failure.

signaller:

- See train (entering or leaving tunnel)
- Signal ’Train in Tunnel’ or ’Tunnel Clear’
- Set or Clear Red flag

Train Driver:

- See Red Flag
- Don’t Enter Tunnel

Seems OK doesn’t it?

This is what happened:

- The first train entered the tunnel, and the signaller sent a 'Train in Tunnel' indication to the other signaller. He (it was a he) went out to set the red flag ..... just as a second train whizzed past.

  The signaller thought about this for a while and then sent a second 'Train in Tunnel' indication.

- At the other end of the tunnel, the signaller received two messages, and then a single train came out of the tunnel. He signalled 'Tunnel Clear', waited another few minutes, and then sent a second 'Tunnel Clear' message.

  Where was the second train?

- Well - the train driver had seen the flag, and decided to be cautious, so he stopped the train and then started backing out of the tunnel.

  Meanwhile, the first signaller thought that all was well and waved on through the next train.

A flaw in a protocol had led to two trains colliding - inside the tunnel - one going forward and one in reverse.

1.2 Recent history

Since the early days of modern computing there has been a steadily increasing need for more sophisticated data transfer services.

In the **1950**s, the early computers were simple machines with 'Single-task' operating systems. They typically had a single console, and data was entered either by cards, or on paper tape.

In the **1960**s, demand for large scale data entry grew, and the 'on-line' batch system was developed. The operating systems were better, and allowed many terminals to be connected to the same computer. These terminals typically would allow deferred data entry - that is the files would not be updated until a 'batch' of data was ready. The data entry was often done during the day, the processing of the data at night.

In the **1970**s, on-line integrated systems were developed. These allowed terminals to access the files, as well as update them. Database technology allowed immediate display of the effect of completed transactions. Integrated systems generated multiple transactions from single entries.

Since the 1980s we have moved to distributed databases and processing. In the 80s and 90s (see figure 1.2), we see large machines - even the workstations have significant computing power. There are many more options for interconnecting the machines.

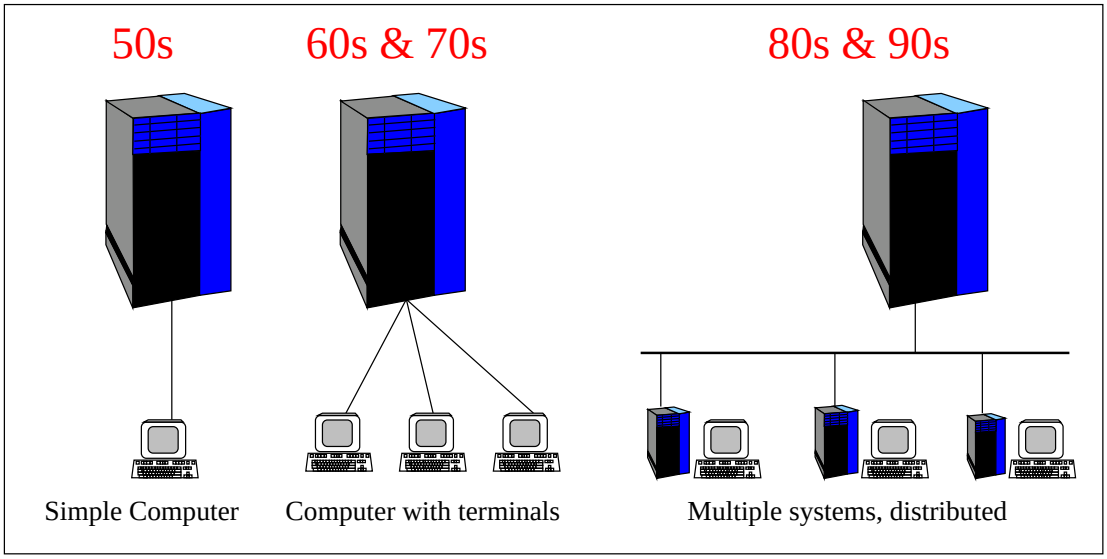


Figure 1.2: Computer development.

> *At this stage we should differentiate between distributed systems and computer networks. A distributed system is a computer network in which the processing may be distributed amongst the computers. The user may not be aware on which computer the software is actually running. A computer network by contrast has a collection of communicating processors, but each users processing is done on a single computer. Modern networks are often mixed.*

The principal *reasons* for computer networks then are resource sharing, saving money and reliability.

The principal *applications* are:

- **remote programs** (rather than multiple copies)
- **remote databases** (as in the banking system, airline reservations)
- **value added communications** (service)

We also have widely varying speed requirements in computer networks, and both tight and loose couplings between machines, and hence there is no one solution to the computer networking problem.

Computer networks range from small systems interconnecting chips on a single PCB[1], up to world wide computer networks. There are many possible ways of connecting the machines together, (topologies), the main ways being the star, ring and bus systems. In general the larger

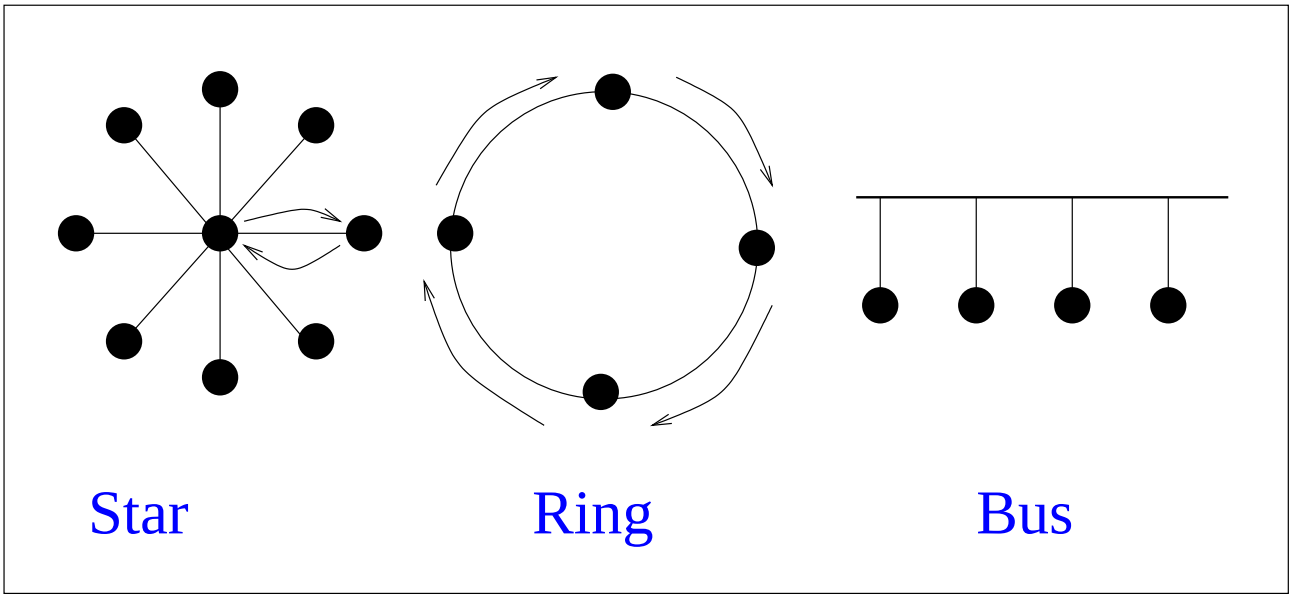


Figure 1.3: Network topologies.

networks are irregular and may contain sections with each of the above topologies. The smaller networks are generally regular.

The largest computer network in the world is the Internet[2] which interconnects over 1,000,000 computers worldwide. The stability, management and topology of this network varies. In some areas, it is left up to University departments - in others there are strong commercial interests.

[1](such as that found on a transputer). A high bit rate serial link between the processors.

[2]Note that you should always capitalize the word *Internet* if you are referring to this 'internet'. If you are referring to any 'interconnected network', you can use *internet*.

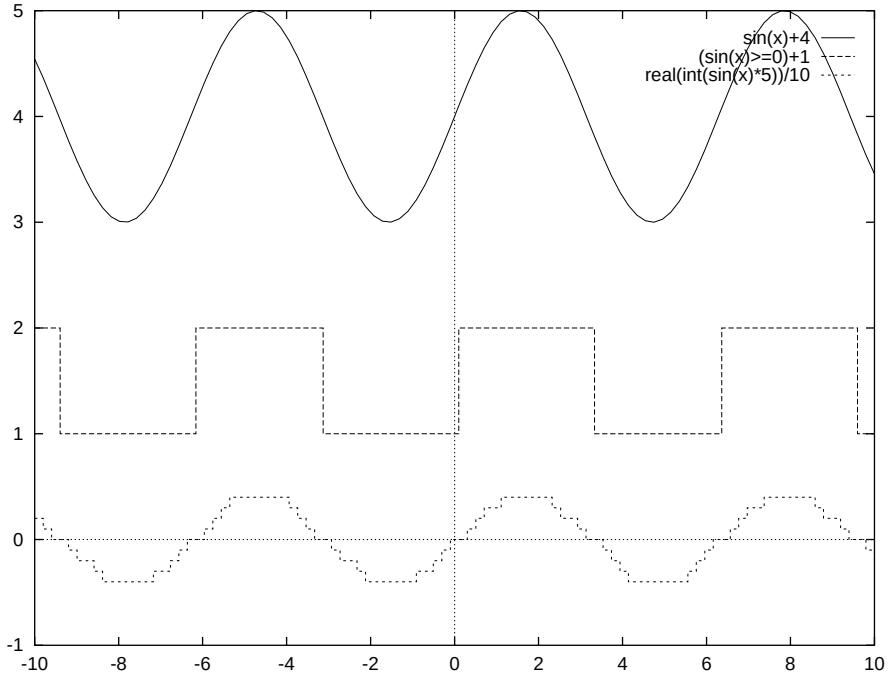


Figure 1.4: Digital and analog Signals.

1.3 Communication theory

When studying the transfer of data generally, there are some underlying physical laws, representations and limits to consider.

Is the data analog or digital? What limits are placed on it? How is it to be transmitted?

1.3.1 Analog & digital

An analog signal is a continuous valued signal. A digital signal is considered to only exist at discrete levels.

The (time domain) diagrams are commonly used when considering signals. If you use an oscilloscope, the display normally shows something like that shown in figure 1.4. The plot is **amplitude versus tim**e. With any analog signal, the repetition rate (if it repeats) is called the *frequency*, and is measured in Hertz (pronounced 'hurts', and written Hz). The peak to peak signal level is called the *amplitude*.

The simplest analog signal is called the sine wave. If we mix these simple waveforms together, we may create any desired waveform. In figure 1.5, we see the sum of two sine waves - one at a frequency of 1,000Hz, and the other at three times the frequency (3,000Hz). The amplitudes of the two signals are 1 and $\frac{1}{3}$ respectively, and the sum of the two waveforms shown

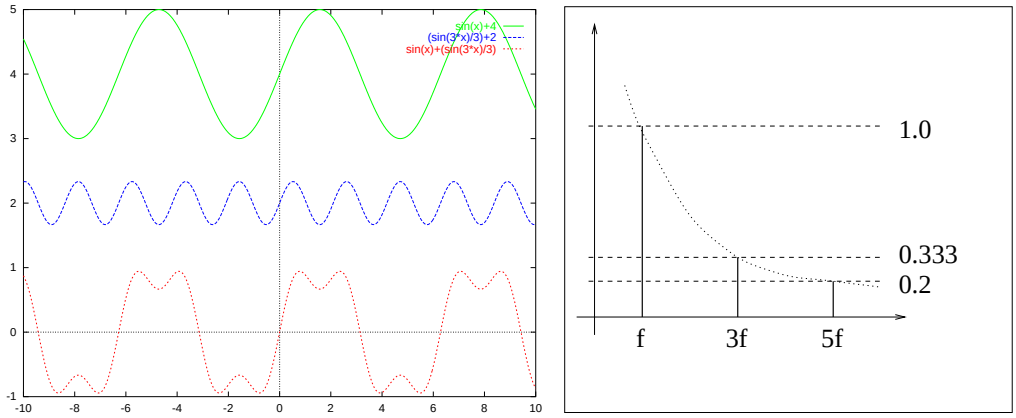


Figure 1.5: Sum of sine waveforms.

below, approximates a *square* wave. If we were to continue summing these waves, in the same progression, the resultant waveform would be a square wave:

$$\sum_{n=1}^{\infty} \frac{1}{n} \sin(2\pi n f) \quad \text{(for odd n)} \Rightarrow \text{a square wave of frequency } f$$

We may also represent these signals by frequency domain diagrams, which plot the amplitude against *frequency*. This alternative representation is also shown in figure 1.5.

1.3.2 Fourier analysis

One way of representing any simple periodic function is in this way - as a sum of simple sine (and cosine) waveforms. This representation method is known as *'Fourier Analysis'* after Jean-Baptiste Fourier, who first showed the technique. We start with the equation for constructing an arbitrary waveform $g(t)$:

$$g(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(2\pi n f t) + \sum_{n=1}^{\infty} b_n \sin(2\pi n f t)$$

f is the fundamental frequency of the waveform, and a_n and b_n are the amplitudes of the sine and cosine components at each of the harmonics of the fundamental. Since a_n and b_n are the only unknowns here, it is easy to see that if we know the fundamental frequency, and the amplitudes a_n and b_n, we may reconstruct the original signal $g(t)$.

For any $g(t)$ we may calculate a_0, a_n and b_n by first noting that the integral over the interval $[0, T]$ will be zero for the summed terms:

$$a_0 = \frac{1}{T} \int_0^T g(t)\, dt \quad \text{where } T = \frac{1}{f}$$

and by multiplying both sides of the equation by $\sin(2\pi kft)$, and noting that:

$$\int_0^T \sin(2\pi kft)\sin(2\pi nft)\,dt = \begin{cases} 0\,for\,k \neq n \\ \frac{T}{2}\,for\,k = n \end{cases}$$

and

$$\int_0^T \sin(2\pi kft)\cos(2\pi nft)\,dt = 0$$

We can then integrate to get:

$$b_k = \frac{2}{T}\int_0^T g(t)\sin(2\pi kft)\,dt$$

Similarly, by multiplying by $\cos(2\pi kft)$, we get

$$a_k = \frac{2}{T}\int_0^T g(t)\cos(2\pi kft)\,dt$$

A bipolar square wave gives $a_k = 0$, and

$b_1 = 1$	$b_2 = 0$	$b_3 = \frac{1}{3}$	$b_4 = 0$	$b_5 = \frac{1}{5}$	$b_6 = 0$ ...

We re-create our waveform by summing the terms below.

$$\tfrac{4}{\pi}(sin(2\pi ft) + \tfrac{1}{3}sin(6\pi ft) + \tfrac{1}{5}sin(10\pi ft) + \tfrac{1}{7}sin(14\pi ft) + ...)$$

In figure 1.6, we see four plots, showing the resultant waveforms if we sum

- the first term
- the first two terms
- the first three terms (and so on...)

As we add more terms, the plot more closely approximates a square wave.

Note that there is a direct relationship between the bandwidth of a channel passing this signal, and *how accurate* it is.

- If the original (square) signal had a frequency of 1,000Hz, and we were attempting to transmit it over a channel which only passed frequencies from 0 to 1,000Hz, we would get a sine wave.
- If the channel passed frequencies from 0 to 3,000Hz, we would get a waveform something like the third one down in figure 1.6.

Another way of stating this is to point out that the higher frequency components are important - they are needed to re-create the original signal faithfully. If we had two 1,000Hz signals, one a triangle, one a square wave - if they were both passed through the 1,000Hz bandwidth limited channel above, they would look identical (a sine wave).

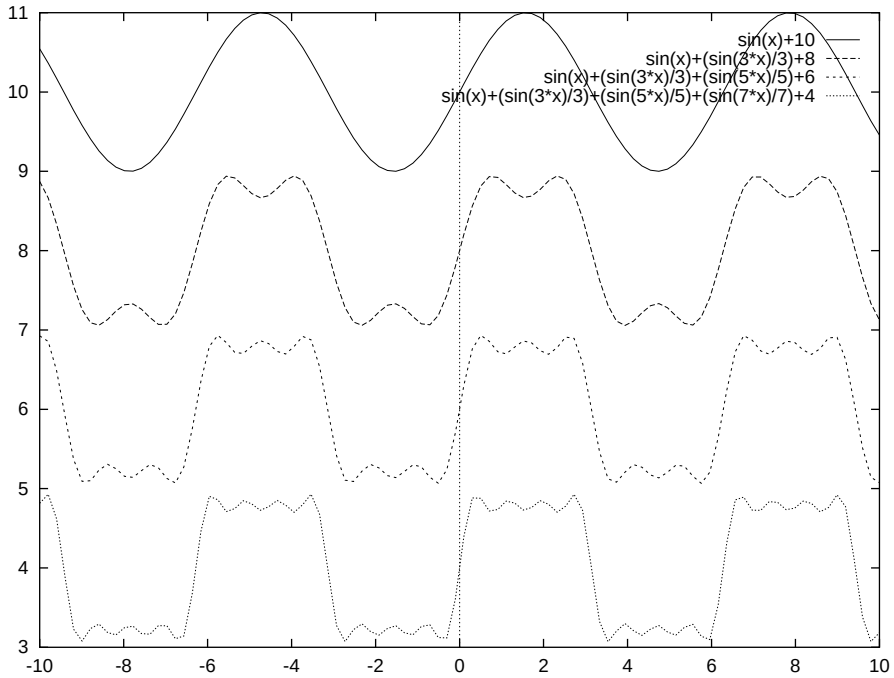


Figure 1.6: Successive approximations to a square wave.

1.3.3 Shannon and Nyquist

Other important relationships found in data communications relate the bandwidth, data transmission rate and noise. Nyquist shows us that the maximum data rate over a limited bandwidth (H) channel with V discrete levels is:

Maximum data rate = $2H\,log_2V$ bits/sec

For example, two-Level data cannot be transmitted over the telephone network faster than 6,000 BPS, because the *bandwidth* of the telephone channel is only about 3,000Hz.

Shannon extended this result for noisy (thermal noise) channels:

Maximum BPS = $H\,log_2(1+\frac{S}{N})$ bits/sec

A worked example, with a telephone bandwidth of 3,000 Hz, and using 256 levels:

$$\begin{aligned} D &= 2 * 3000 * (log_2 256)\,bps \\ &= 6000 * 8\,bps \\ &= 48000\,bps \end{aligned}$$

But, if the S/N was 30db (about 1024:1)

$$\begin{aligned} D &= 3000 * (log_2 1025)\, bps \\ &= 3000 * 10\, bps \\ &= 30000\, bps \end{aligned}$$

This is a typical maximum bit rate achievable over the telephone network.

1.3.4 Baseband and modulated signals

A baseband signal is one in which the data component is directly converted to a signal and transmitted. When the signal is imposed on another signal, the process is called modulation.

We may *modulate* for several reasons:

- The media may not support the baseband signal
- We may wish to use a single transmission medium to transport many signals

We use a range of modulation methods, often in combination:

- Frequency modulation - frequency shift keying (FSK)
- Amplitude modulation
- Phase modulation - phase shift keying (PSK)
- Combinations of the above (QAM)
- In the section on modems (section 3), we will discuss these methods in more detail.

1.4 Media

The transmission medium is just the medium by which the data is transferred. The type of medium can have a profound effect. In general there are two types:

- Those media that support point to point transmission
- Multiple access systems

One of the principal concerns is '*how to make best use of a shared transmission medium*'. Even in a point to point transmission, each end of the transmission may attempt to use the medium and block out the other end's use of the channel. There are several well known techniques:

- **TDM** - Time Division Multiplexing
- **FDM** - Frequency Division Multiplexing
- **CSMA** - Collision Sense Multiple Access
- **CSMA/CD** - CSMA with Collision Detection

Here are some sample mediums for comparison:

MAGNETIC TAPE. (Point to point)

If you wanted to transmit 180MB (megabytes) across town to another computer site , and you only had a 9,600 bytes/sec line to the other site, you could only transmit 1,000 characters a second there, and the whole transfer would take 180,000 seconds (3,000 minutes/50 hours). The whole process could be much more efficiently performed by copying the data to a single nine track magnetic tape, sticking it in a courier pack and sending it across town. *There is always more than one way to skin a cat.*

TELEPHONE NETWORK. (Point to point)

The telephone network provides a low bandwidth (300Hz to 3kHz) point to point service, through the use of twisted pair cables.

There are some problems with the telephone network:

Echo suppressors - Put on long lines to stop line impedance mismatch echoes from interfering with conversations. These echo suppressors enforce one-way voice communication (half duplex), and inhibit full duplex communication. The echo suppressors may be cut out by a pure 2,100 Hz tone.

Switching Noise - The telephone network is inherently noisy with lots of impulse noise (Switch/relay pulses and so on). A 10mS pulse will chop 12 bits out of a 1,200 bits/sec transmission.

Limited Bandwidth - An ordinary telephone line has a cutoff frequency of 3,000Hz, enforced by circuitry in the exchanges. If you attempt to transmit 9,600 bps over these lines, the waveform is dramatically changed.

COAXIAL CABLE. (Point to point or Multiple access)

Two kinds of coax are in common use - 50Ω[3] for digital signalling and 75Ω for analog (TV) transmission. The digital coax has high bandwidth and good noise immunity. Bandwidths of 10Mbps are possible at distances over 1 km (limited of course by the Shannon limit given in Section 1.3.3).

[3]We use the greek letter Ω (ohm) as the unit of resistance.

FIBRE OPTICS. (Generally point to point)

Fibre optic technology is rapidly replacing other techniques for long distance telephone lines. It is also used for large traffic networks, as current technology can transmit data at about 1,000Mbps over 1km. The FDDI[4] standard specifies 100Mbps over 200km. The Shannon limit is much higher than this again, so there is room for technological advancement.

RADIO. (Multiple Access)

Radio is often the transmission medium for digital data transmission. For example, when NASA were looking for 'sandbags' to go in their test rocket launching systems, enterprising HAM radio enthusiasts asked if they could provide the weight, and designed small satellites to retransmit their signals. There are now 12 HAM satellites in orbit, mostly retransmitting digital data between HAMs all over the world. The HAMs use a data transmission protocol called AX.25 which is similar to X.25 - a widely used standard. The difference is mostly in the address fields, which allow 'HAM' call signs to be inserted.

Any HAM in the footprint of the satellite may transmit to it and receive from it, and so HAMs use various protocols to minimize collisions.

[4]Fibre Optic Distributed Data Interface

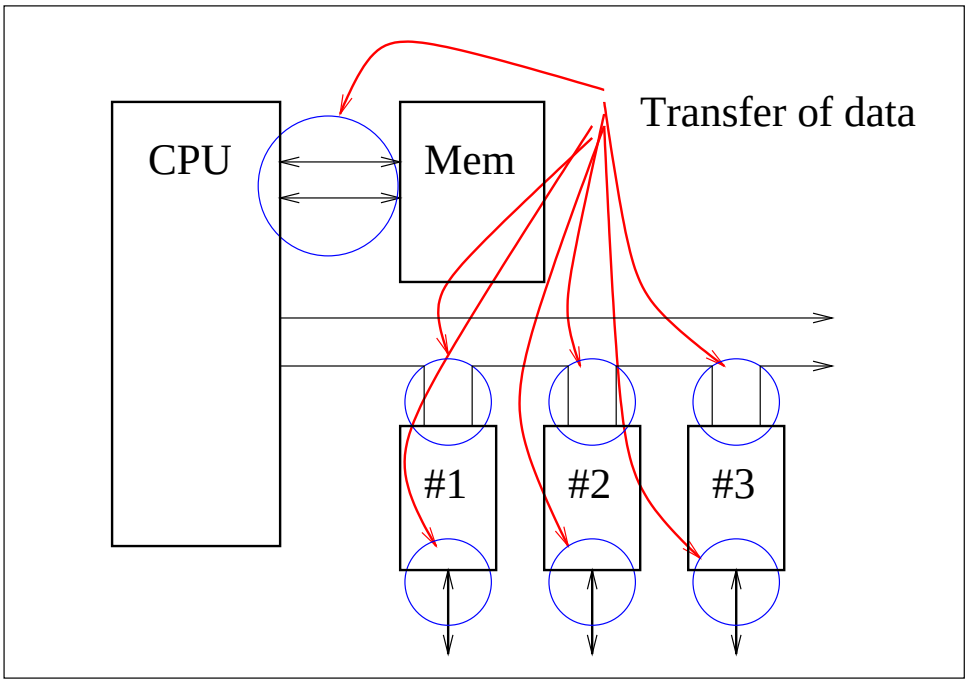


Figure 1.7: Model of computing.

1.5 Computer hardware

One of the interesting features of the study of computer communications is that it is relevant in virtually all levels of computer use - from *how computer chips on a circuit board interact* to *how databases share information.*

If you look at the back of a PC or a MAC (or a Sun or an SGI!), you find various connectors and cables. Some just supply power to the unit, but most of the others involve transfer of data.

In this section we take an initial look at some of the communication schemes found in a computer. The circled elements in figure 1.7 indicate (possibly different) data transfer standards.

- Parallel port
- Serial port
- Keyboard port
- Modem
- Backplane
- SCSI port
- MAC bus
- The Monitor cable

1.5.1 Backplane & IO busses

The backplane of the computer (which is sometimes the same as the I/O bus) is the physical path that interconnects the processor CPU chips and the memory or I/O devices. It normally has a

large number of *address*, *control* and *data* lines. The voltages found on the backplane are just the normal chip operating voltages (3.6V to 5V), the speed is high, the distances involved are small, and the data is digital.

Since the backplane is a bus, we have to be careful that only one chip at a time tries to use it. If chips conflict when trying to transfer data it is called **contention**. Some of the control wires in the bus are used to help resolve contention. We also need handshaking signals to regulate the flow of data. The receiving chip/board must be able to *slow down* the flow of data from a faster chip/board.

IBM PCs may have several bus systems, for memory (the memory bus), video (such as VESA local bus) or I/O cards (the backplane - ISA, EISA,PCI).

PCI is a 64 bit interface in a 32 bit package. The PCI bus runs at 33 MHz and can transfer 32 bits of data (four bytes) every clock tick. A clock tick at 33 MHz is 30 nanoseconds, and memory only has a speed of 70 nanoseconds. When the CPU fetches data from RAM, it has to wait at least three clock ticks for the data. By transferring data every clock tick, the PCI bus can deliver the same throughput on a 32 bit interface that other parts of the machine deliver through a 64 bit path. The PCI bus is used in Power Macintosh systems and PowerPC machines, as well as PCs.

With backplane systems like this, we measure the speed in Bytes/sec.

Example: The PCI bus has a 32 bit (4 byte) data transfer size, and a 30nS cycle time, and so the transfer speed is **133MB/sec.**

Note: if our backplane was 64 bits wide, with the same cycle time, it would have a transfer speed of **266MB/s.**

By contrast, the **SGI O2** workstation has a unified memory/IO bus with a transfer speed of **2.1GB/sec.**

1.5.2 Parallel port

The parallel port is often labelled as a *printer* port on a PC. These are typically 8-bit ports, with handshaking to support a uni-directional point to point link.

With this data communications link, it is easy to transfer 8-bit data at a relatively slow speed (relative to the backplane). The typical maximum transfer speed is 1Mbps, over 1 to 10 metres. It is common to represent character data using the ASCII character set, which is a 7-bit character encoding method, with an extra bit often added as an error/parity check.

Note: there are other encoding schemes such as EBCDIC (used on IBM mainframes), and 16 bit schemes to support non-english fonts (Kanjii and so on).

In figure 1.8, we see the handshaking for data transfer using the common centronics parallel interface.

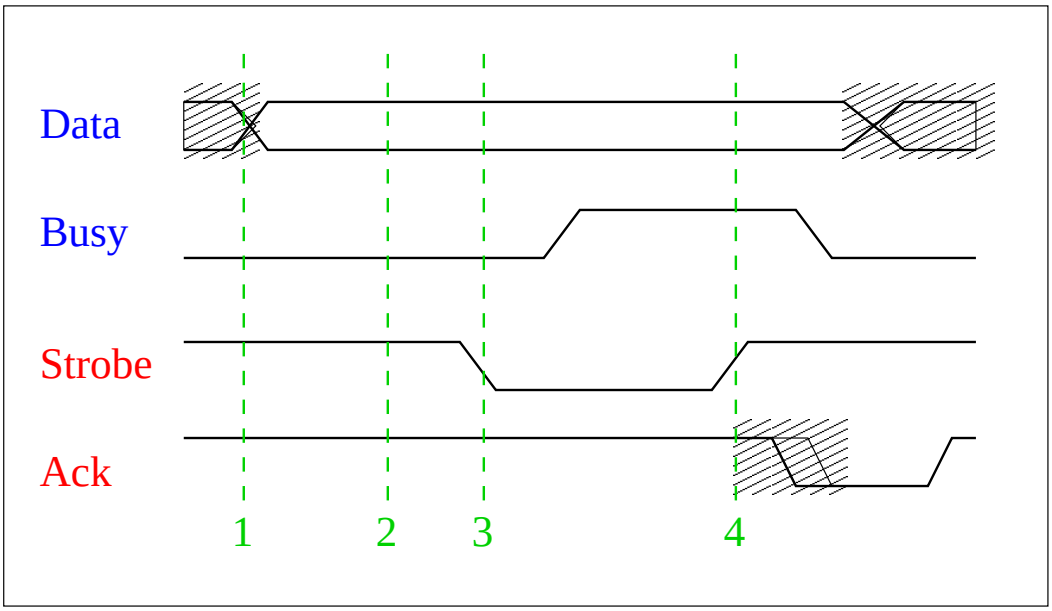


Figure 1.8: Centronics handshake signals.

1. Write the data to the **data register**
2. Read status register to see if printer is **busy**
3. Write to control register to assert the **strobe** line
4. De-assert the **strobe** line

The current parallel standard is IEEE 1284-1994, which provides for a high speed bi-directional transfer of data (50 or so times faster than the old centronics standard), and can still inter-operate with the older hardware in a special compatibility mode.

1.5.3 Serial port

The PC serial port provides two 1-bit point to point data links, with a large number of handshaking wires (as many as are needed for parallel ports). Since the channel width is only 1 bit, the data transfer speed is typically slow (1,200 to 38,400 bits/sec).

Forty years ago, the maximum speed of an electronic printer (teletype) was about 10 (7-bit) characters per second. Data sent to the printer at 75 bits/sec would keep it running continuously. As printers increased in speed, the speed doubled as needed - 75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 and so on. These rates are the common rates supported by serial ports.

When we send data using serial ports, the sender and receiver must be synchronized. This may be done in one of two ways:

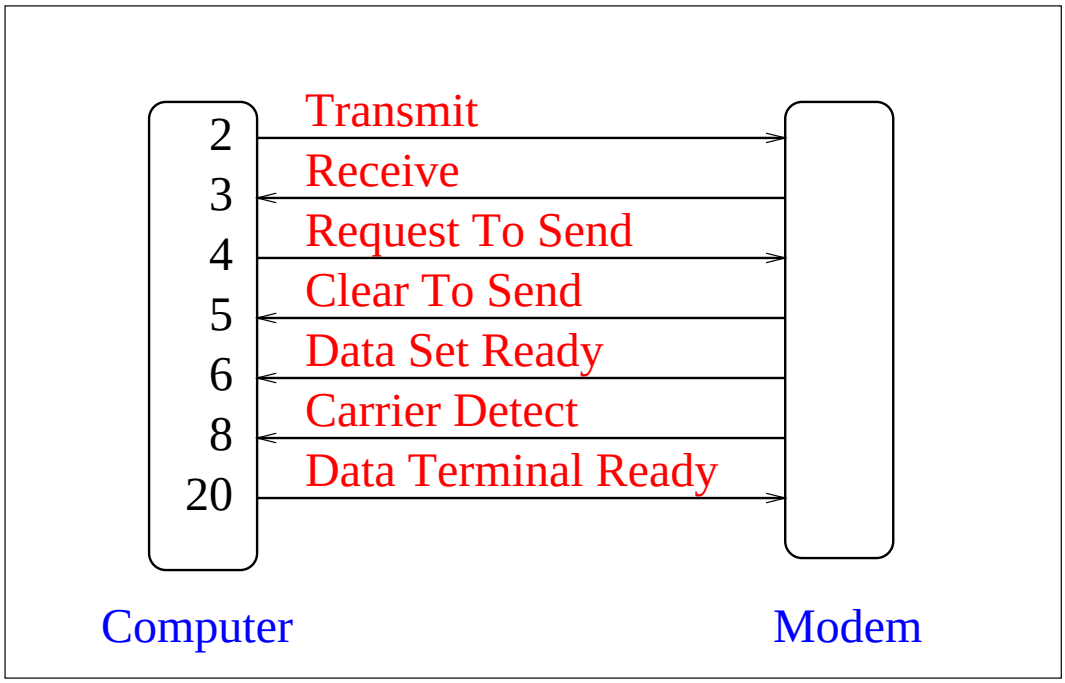


Figure 1.9: RS232-C serial interface.

- Use a synchronizing signal or clock (called **synchronous** transmission)
- Agree on a transfer speed, and wait for a beginning signal (called **asynchronous** transmission)

Most PC serial ports support the RS232-C signalling conventions, but MAC and UNIX workstations often use RS432 standard, which can *look like* RS232 in certain circumstances.

Figure 1.9 shows the main signals found in the RS232 interface.

1.5.4 Keyboard

The PC keyboard is connected to the PC with a 5-wire cable. Two of the wires are power and ground of course, and the other signals on the cable are at TTL (5V) levels,

The other signals provide a bi-directional synchronous serial link between the computer in the keyboard, and the PC. This communication link has a purpose-specific protocol.

1.5.5 SCSI port

Many modern computers use the SCSI (Scuzzy) interface for interconnecting devices, particularly disks, and measuring instruments. the original SCSI interface provided for an 8-bit wide

transfer of data over a 10m cable at 5MB/sec, but modern SCSI-2 and SCSI-3 interfaces provide faster transfers. On an SGI O2, the disks use a 40MB/sec SCSI interface. Transfers occur 32 bits at a time.

The SCSI interface transfers of data are similar to those found in the parallel interface met before. However the interconnection method is different. SCSI is a multimaster bussed scheme, with addressed nodes and defined protocols for handling contention and transfers.

1.5.6 Macintosh LLAP

Macintosh computers come standard with a network system called LLAP[5]. The *localtalk* cabling scheme can vary, and even the connectors on the back of a MAC have changed over the years[6]. A transformer-decoupled multidrop scheme is common, with a data transfer rate of 200kb/sec. The signals are RS422-like.

Localtalk LLAP messages all belong to well-defined protocols specified by Apple Computer Inc, and placed in the public domain.

1.5.7 Monitor cable

The monitor cable is not really a data-communications cable in the same terms as these other computer interfaces, because there is no *'communication'* involved (only data *transfer*). However there are still some points of interest to be found - firstly the amount of data transferred, and secondly the formats or encodings used.

Data Transfer Rate:

With a typical computer screen, we may have 1280*768 pixels. Each pixel may be represented by a 24 bit (3 byte) number to identify its colour. (255 levels for each of red, green and blue). Therefore we require 2,949,120 Bytes to display one frame of our display. If our screen is refreshed 70 times per second (half a frame each time for an interlaced display), we require 98.4MB/sec (787.2Mb/sec) to be transferred from the video display card to the display hardware. This is a continual demand, but we don't mind errors - they just give temporary aberrations to the display.

Encoding:

There are several standards, but commonly use separate cables for each of the red, green and blue components. each of these cables provides an analog signal representing the intensity of the colour at any instant in time.

[5] Localtalk Link Access Protocol.

[6] Originally a DB9 connector - but now a mini-din8 connector is used.

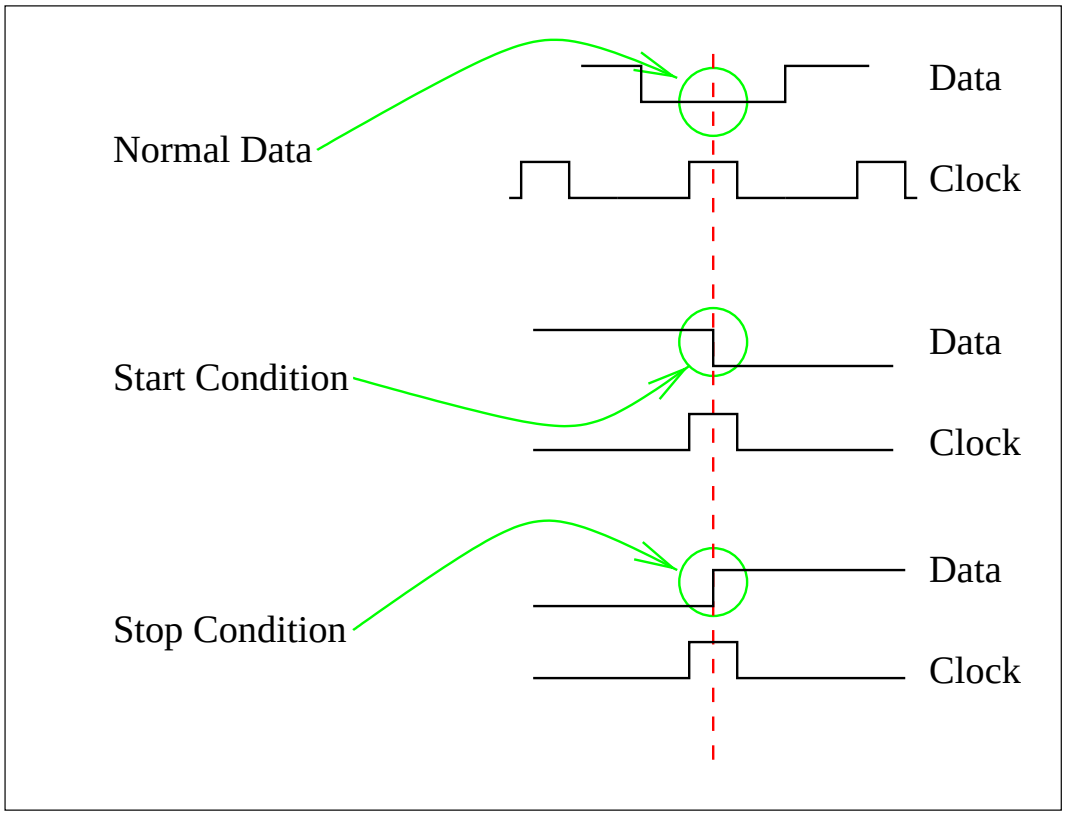


Figure 1.10: I^2C synchronization.

Another cable may contain (digital) synchronizing data.

1.5.8 I^2C

I^2C is a standard for **i**nterconnecting **i**ntegrated **c**ircuits on a printed circuit board. It has a synchronous bus multi-master scheme, with a well developed addressing scheme. Only 3 wires are needed to interconnect the ICs, simplifying circuit construction and chip count.

To ensure synchronization, I^2C indicates the beginning and end of data with special signals as shown in figure 1.10.

1.6 Standards organizations

The major standards organisations display an amazing conformity in their standards, mostly because they tend to be rubber stamp organisations. However manufacturers do modify their equipment to conform to these standards.

> *The nice thing about standards is that you have so many of them. Furthermore, if you do not like any of them, you can just wait for next year's model.*

- **The National Bureau of Standards** is an American Federal regulatory organization which produces the Federal Information Processing Standards (FIPS). Their standards must be followed by all manufacturers producing equipment for the US government and its agencies (except defense).
- **The Electronic Industries Association** is an organization made up of manufacturing interests in the US. They are responsible for RS232 and similar standards.
- **The Institute of Electrical and Electronic Engineers** is a professional organization of engineers. They prepare standards in a wide range of specialities.
- **The American National Standards Institute** is an umbrella organization for many US standards organizations. Accreditted member organizations submit their standards for ANSI acceptance.
- **The International Organization for Standards** (**ISO**) generate standards covering a wide range of computer related topics. The U.S. representative is ANSI.
- **The Consultative Committee on International Telephone and Telegraph**. Its standards are law in most member countries with nationalised telecommunications (much of Europe). The US representative is the Department of State. The standards mostly relate to telephone and telecommunications.

Many of the standards adopted were first fixed on by committee workers in national standards organisations. The workers are often provided by (competing) companies, who sometimes don't agree on a 'standard'. The traditional ISO technique to deal with this situation is to create multiple (incompatible) standards. A clever extension to this is to make these standards different from the original company standards.

> **For example:** three manufacturers (Xerox,GM & IBM) submitted three different LAN standards to IEEE, who produced three different standards (IEEE 802.3, 802.4 and 802.5). Each of these standards has slight format differences from the other, for no valid reason except perhaps that no-one wanted to offend anyone else. These

> standards were adopted by ISO (ISO 8802.3, 8802.4, 8802.5) and have three incompatible formats. When ISO were pressured to construct a standard to 'bridge' these incompatible formats, they did slightly better, and came up with two incompatible bridges!

Another radical ISO technique is to leave out anything that is controversial. ISO spent much time discussing data security and encryption, and finally decided that it should be left out, not because it was unimportant, but because it caused too many arguments.

Chapter 2

OSIRM

It is against this background that we present the ISO OSI (Open Systems Interconnect) model. This model is dated, but provides a framework to hang your understanding of computer networking on. It is modelled on IBMs SNA (System Network Architecture) but with differences.

What are the advantages of this?

- We can change layers as needed, as long as the interface remains constant. The service and the protocol are decoupled, and hence other level protocols may be changed without affecting the current level.
- the network software is easier to write, as it has been modularised.
- Client software need only know the interface.

Each layer is defined in terms of a **protocol** and an **interface**. The protocol specifies how communication is performed with the layer at the same level in the other computer. This does not for instance mean that the *network* layer directly communicates with the *network* layer in the other computer - each layer uses the services provided by the layer below it, and provides services to the layer above it. The definition of services between adjacent layers is called an **interface**

Remember -

- a **PROTOCOL** specifies the interaction between peer layers,
- an **INTERFACE** specifies the interactions between adjacent layers of the model.

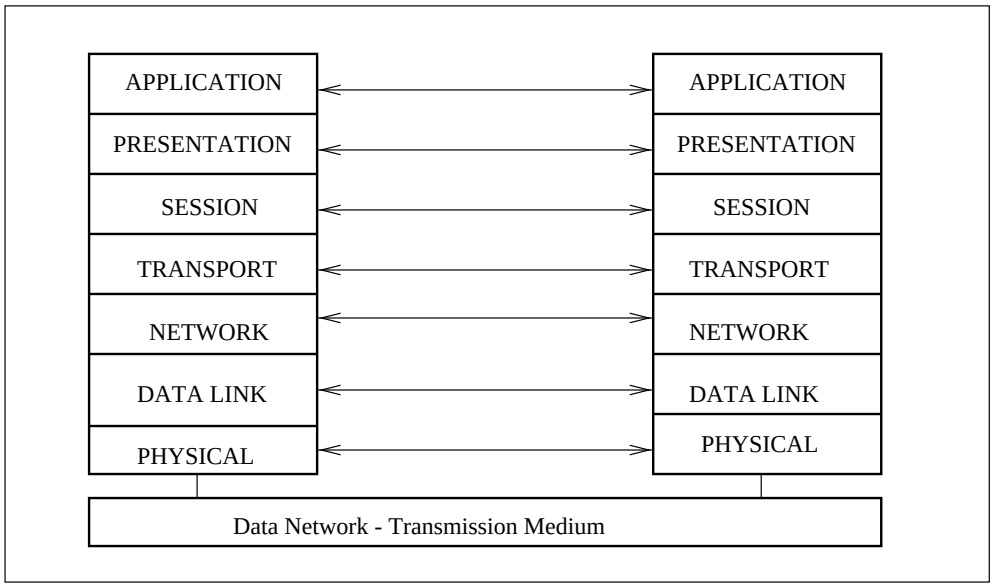


Figure 2.1: Layering in the ISO OSI reference model.

2.1 The layers

PHYSICAL LAYER

The physical layer is concerned with transmission of data bits over a communication line - that is the transfer of a '1' level to the other machine. In this layer we worry about connectors, cables, voltage levels etc.

DATALINK LAYER

The datalink layer is concerned with the (hopefully) error-free transmission of data between machines. This layer normally constructs frames, checks for errors and retransmits on error.

NETWORK LAYER

The network layer handles network routing and addressing. It is sometimes called the packet switching network function. At this level are performed line connection and termination requests. X.25 is the internationally accepted ISO network layer standard. IP is the Internet network layer standard.

TRANSPORT LAYER

The transport layer provides an interface between the 'network communication' layers (1..3) and the higher service layers. This layer ensures a network independent interface for the session

layer. Since there can be varying network types, ISO provides for five levels of 'Class of service' ranging from a simple connection manager (class 0) to a full error and flow control manager (class 4). You would use class 0 when communicating over a PSDN, and class 4 when using a PSTN.

The Internet protocol suite has two common transport protocols:

- **TCP** - Transmission Control Protocol
- **UDP** - User Datagram Protocol

The transport layer isolates the upper layers from the technology, design and imperfections of the subnet.

SESSION LAYER

The session layer is closely tied to the TRANSPORT layer (and often the software is merged together). The session layer is concerned with handling a session between two end users. It will be able to cleanly begin and terminate a session, and provide clean 'break' facilities.

PRESENTATION LAYER

This layer is concerned with the syntax of the data transferred. For example it may be desired to convert binary data to some hex format with addresses. The conversion to and from the format is handled at the presentation layer. Data may also be encoded for security reasons here.

APPLICATION LAYER

The application layer provides the user interface to the network wide services provided. For example file transfer, electronic mail. Normally this layer provides the operating system interface used by user applications.

2.2 Example

Let us say that the *President* of one nation decides to declare war on another nation. The *President* calls in the *Secretary of State* who handles all such matters. The *Secretary* decides that the appropriate way to handle this is to announce the fact at the United Nations, and so calls in the country's *U.N. representative*. The *U.N. representative* rushes off to the U.N. and after gaining the floor announces 'WAR'!!!

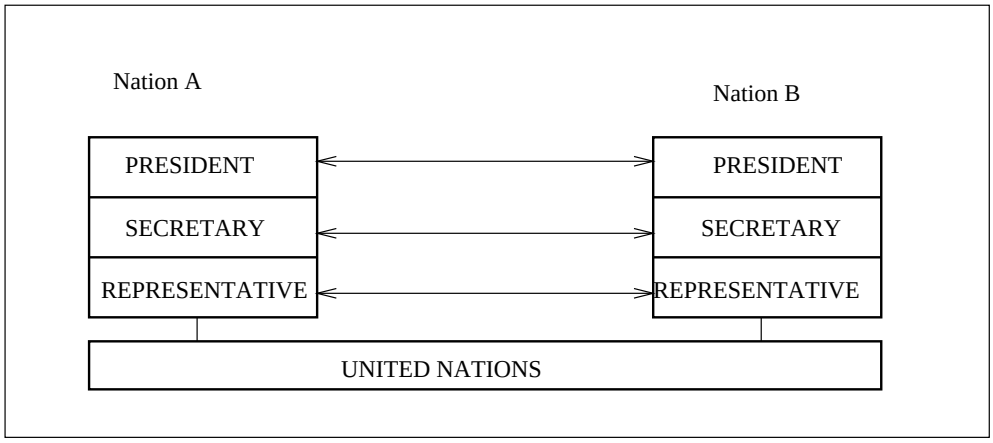


Figure 2.2: War is declared!

The *U.N. representative* of the opposing nation stalks out, and calls her *Secretary of State* who in turn warns the other *President*.

In this example, the WAR 'challenge' is from *President* to *President*, although the communication is down through the protocol stack. Note also that the *secretary* may have decided that an appropriate way to issue the challenge may have been to drop bombs on the capital city of the opposing nation, bypassing the slow technique given above.

There may also have been more levels than given above - for example the challenge may need to be translated, or the 'representative' may have decided that the best way to communicate the challenge was from secretarial staff to secretarial staff.

Why have we chosen a military example above?

> *Most of our knowledge about computer networking comes from ARPA (Advanced Research Projects Agency), a branch of the U.S. military, which was set up in the early 1960s and did research into computer networking. The ARPA internet has been running since the late 1960s. The ARPA internet has no session or presentation layers, as no one has found a use for them in the the first 25 years of operation. One ARPANET transport protocol is called TCP (for Transmission Control Protocol). The ARPANET network protocol is called IP (for Internet Protocol). TCP/IP is in widespread use throughout the computer world.*

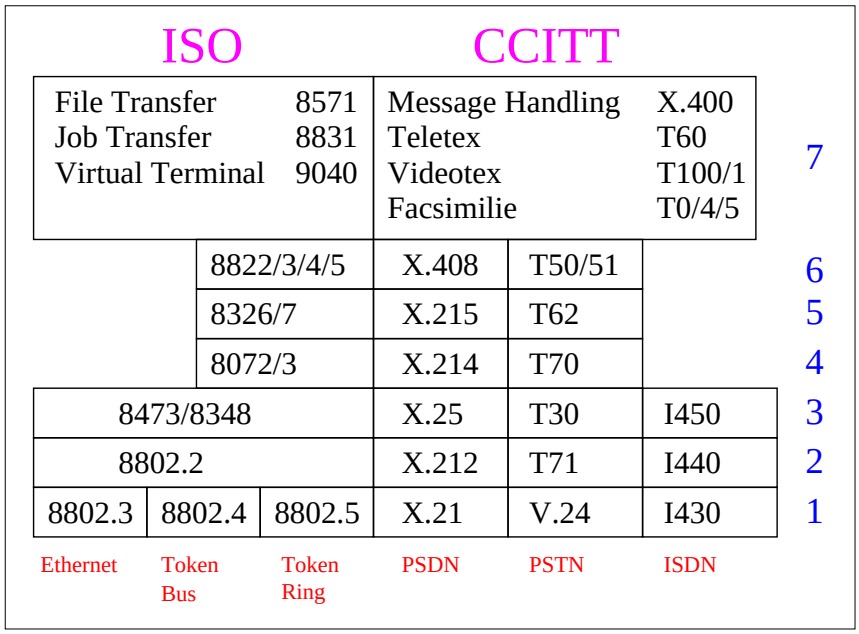


Figure 2.3: ISO and CCITT protocols.

2.3 Sample protocols

In figure 2.3, we see an assortment of protocols. On the left are some ISO standard protocols - on the right some CCITT ones. The protocols are arranged by the layer in which they appear. At layer 3, the network layer, we see the CCITT standard X.25. At the physical layer we see standards such as V.24 (found in modems).

Notice that the CCITT naming standard allows you to *take a guess* as to its use:

- the X.xx standards are used in the PSDN
- the V.xx and T.xx standards are used in the PSTN
- the I.xxx standards are for ISDN

Chapter 3

Layer 1 - Physical

Definition:

> *The physical layer is concerned with the transmission of bits through a medium.*

In the following chapters, we describe important features of each of the layers of the OSI reference model. Each chapter follows a common format:

1. Introduce the layer
2. Give some sample protocols
3. Outline the addressing schemes
4. Sections on the characteristics and modes of operation
5. Diagnostic tools for this layer

3.1 Sample 'layer 1' standards

In table 3.1, we summarise various signalling standards related to computer communications.

They range in speed from 38.4Kb/s to 125Mb/s (a range of about 2,500:1). The maximum design distances range from 10m to 200km (a range of about 20,000:1).

All are in common use at present, and are considered adequate in their application area:

- **MAC, Token ring, Ethernet & Spread-Spectrum** - used to *network* computers on LANs (Local Area Networks).

Standard	Method	Speed	S/P	Dist	Type	Cable	Conn.	Isol.
RS232	+/- 12v	38.4Kb/s	S	100m	P to P	Various	DB25	N
				100m	P to P	Various	DB9	N
MAC	+/-1v	220Kb/s	S	1.5km	M/drop	Various	Various	Y/N
Centronics	0,+5v	800Kb/s	P	10m	P to P	Ribbon	Special	N
SCSI	0,+5v	40Mb/s	P	10m	M/drop	Ribbon	Special	N
Ethernet	0,-1.2v	10 Mb/s	S	90m	P to P	UTP	RJ11/RJ45	Y/N
				185m	M/drop	RG58	BNC	Y/N
				500m	M/drop	RG11	Vampire	Y/N
Fibre (FDDI)	Light	125Mb/s	S	200km	ring	Fibre	Various	Y
Token Ring	0,+5V	16Mb/s	S	1.5km	ring	Various	Various	N
S. Spect.	Radio	2Mb/s	S	2km	M/drop	-	None	Y

Table 3.1: Sample physical layer standards.

- **RS232** - for linking two machines, connecting to a modem, or connecting to a printer.
- **SCSI** - commonly used for connecting to disks.
- **FDDI** - for linking campus-wide networks (MAN - Metropolitan Area Networks)
- **Centronics** - for connecting directly to printers from a computer.

3.1.1 Token ring cabling

With *token ring* cabling systems, there are quite a few different cabling types used. Most of them use a setup that looks like a star. The actual wiring is a ring laid out as a star. At the center of the star is a MAU[1]. The MAUs come in all sorts of flavours - from systems with no active components (just make before break plugs) through to computer driven, monitored hubs.

3.1.2 Localtalk cabling

All Macs come with networking hardware and software built in. Older style Macintosh computers have a 200kb/s system with a datalink layer like HDLC[2]. Again there are multiple cabling schemes. There are also two different plugs (DB9 and MINI-DIN8). The signals that come out are RS422 like, and are not isolated. For this reason, most Mac cabling schemes use a *drop box* to connect to a network. In the *drop box* is an isolation transformer. The system will support 100 machines over 1500 m.

[1] Media Access Unit
[2] High-level Data Link Control

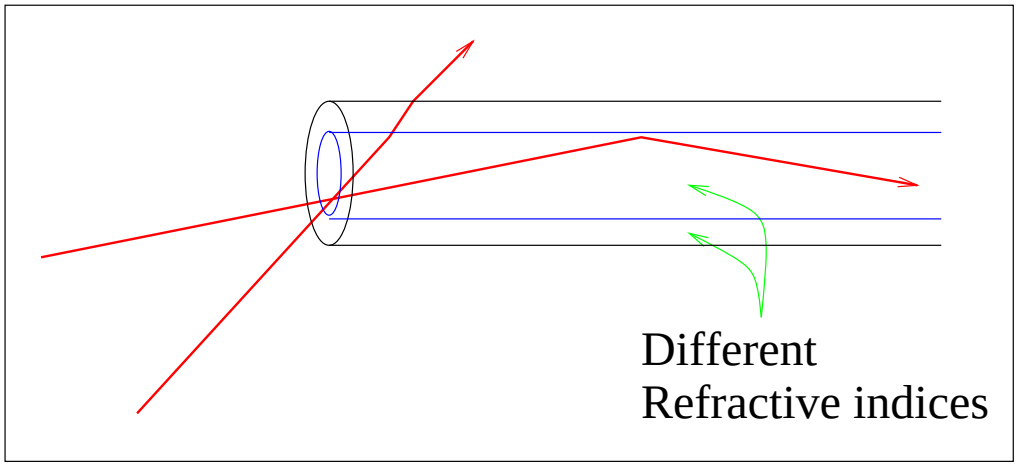


Figure 3.1: Total internal reflection in a fibre.

3.1.3 UTP cabling

UTP[3] is used for many systems (Token ring/Mac etc), but principally for 10M and 100M ethernet.

The original idea was to use existing telephone wiring over distances of 30 - 90 meters. However, now people are replacing all of their building wiring with special UTP cable (known as CAT5) to allow their network to work at 100Mbps at some time in the future (if they buy 100Mbps hardware). When you use these building wire systems, it is called *future proofing* the network. The CAT5 cable will work at 100 Mbps over 90 meters.

The general architecture at the physical layer is STAR. For ethernet (CSMA/CD[4]) networking over UTP, the wires are patched to either a *hub* or an *etherswitch.*

3.1.4 Fibre optic cabling

Fibre technology involves transmitting modulated light down a small plastic or glass fibre. The fibre is made from two different materials, and if the incident light reflects from the material interface at a small enough angle, the *light* reflects totally. If the angle is increased there is not much reflected. Fibre cannot be sharply bent for this reason, and also to prevent damage to the fibre.

Fibre cannot (easily) be tapped, so it is only suitable for point to point cabling. Any intrusion into the fibre generally stops it working.

FDDI - (The Fibre Distributed Data Interface) is a 100 Mbps technology, a little like token ring. It has a 200km maximum length. The cabling may not *look* like a ring (by using multiple fibres in one cable).

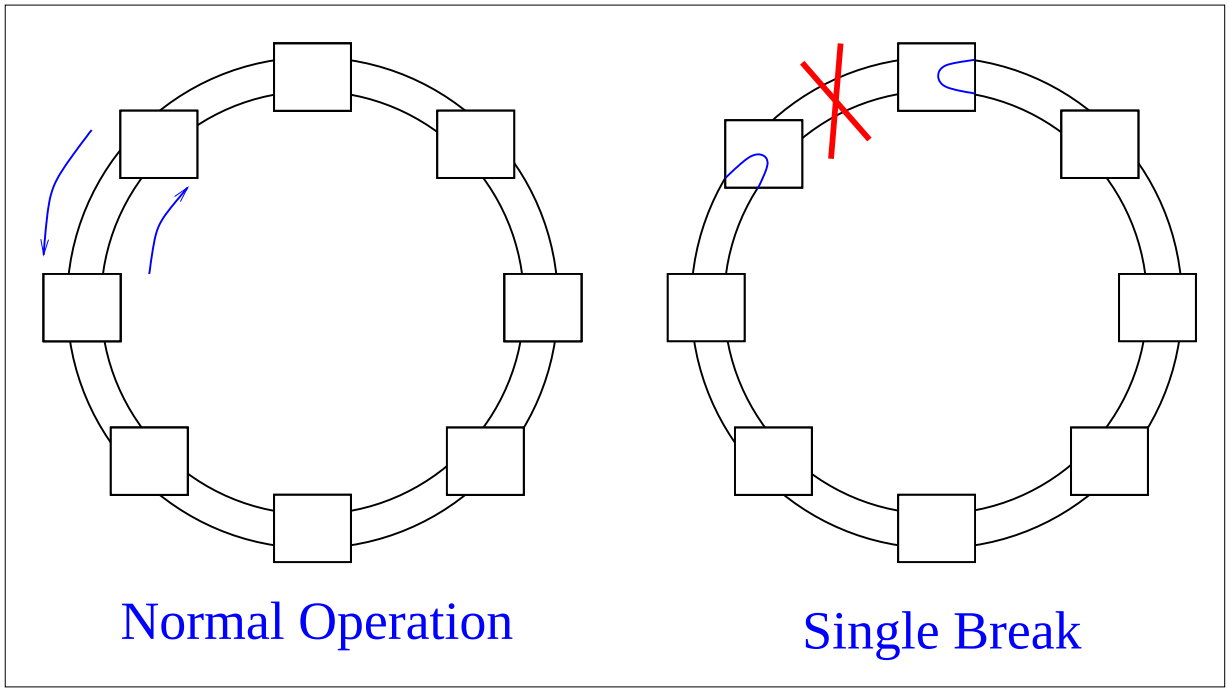


Figure 3.2: Counter rotating rings in FDDI.

Note: FDDI uses dual counter rotating rings. When a cable or its interface is damaged, the interface on either side detects this and heals the ring. It is possible to end up with 2 or more rings.

Fibre is also used for interconnecting ethernets. A standard to do this is FOIRL[5].

3.1.5 *Thin* and *thick* ethernet cabling

One common standard for reticulating ethernet is called *thin ethernet*. You may also see the term *10base2*. (The *10* implies *10Mbps*, *base* implies *baseband*, and *2* implies that you can have about *200m* in a single segment - actually 185m). The cable connectors are **BNC**, with **T** connectors to connect to the computer, and link together cable segments.

The cable is about 3mm in diameter, has a characteristic impedance of 50Ω (see section 3.4), uses a multi-drop (or bussed) scheme, and should be labelled RG58.

A better and more expensive standard is *thick ethernet*. The cable is RG11, and typically costs five times the amount of thin ethernet. Thick ethernet is also called 10base5 (the *5* implies that you can have about *500m* in a single segment).

[3]Unshielded Twisted Pair.
[4]Carrier Sense, Multiple Access, with Collision Detection
[5]Fibre Optic Intra Repeater Link

Type	Cable	Length	Machines	Type	Cost
10base2	RG58	185m	30	Bus	low
10base5	RG11	500m	100	Bus	high
UTP	CAT5	90m	2	P/P	low

Table 3.2: Comparison of ethernet cabling strategies.

The cable runs for thick ethernet are unbroken (i.e. we do not cut the cable). Instead, a *'vampire tap'* clamps around the cable and a needle connects with the center conductor. There is often some signal conditioning hardware in the vampire tap box and a drop cable to a 15 pin (AUI[6]) connector on the computer.

3.2 Addressing

At the physical layer, we use the following *manual* methods:

- Tags and markers for connectors.
- Tags and markers for cables.
- Network diagrams.

There is no one standard, but cable suppliers supply cable and connector labelling equipment.

It is not possible to overemphasize the importance of correct labelling for cabling. Most *network* problems have simple solutions along the lines of:

- "Put the plug back into the computer", and:
- "Why did you cut the cable?", along with its companion:
- "Where do you think the cable is broken?"

Clear network diagrams, and marked cables can minimize these sorts of problems.

3.3 Spectrum

We use only a part of the electromagnetic spectrum. In figure 3.3 we see the most common application areas of modulated signalling.

[6]Attachment Unit Interface

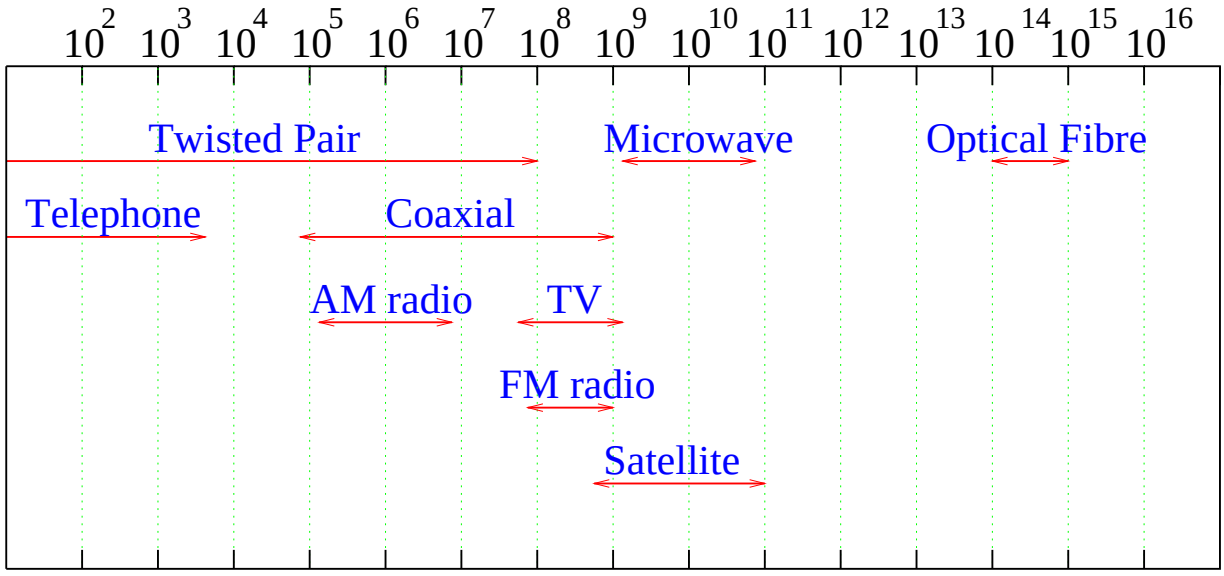


Figure 3.3: Electromagnetic spectrum.

Note: Figure 3.3 has a logarithmic scale, each division encompassing ten times the range of the previous one. So even though the *microwave* part of the spectrum encompasses two divisions (a range of about 100:1), the *optical fibre* part is much larger.

- The bandwidth of *optical fibre* is about 900,000,000 MHz
- The bandwidth of *microwave* is only about 100,000 MHz
- The bandwidth of *coaxial cable* is only about 1,000 MHz

It is clear from Shannon and Nyquist (ref section 1.3.3) that the bit rate on an optical fibre could be much higher than any of the other media given here, if the noise levels are comparable. In fact the noise levels are typically much lower than those found in other media.

3.4 Signals and cable characteristics

If we look at a real signal, it does not always look like the original transmitted signal. There are two factors causing the sort of degradation shown in figure 3.4.

1. The **frequency and gain characteristics** of the cable/media
2. The **group delay** of the cable/media on the signal. Group delay is a term used to indicate the effect caused by the variation in velocity of the different frequency components of the signal.

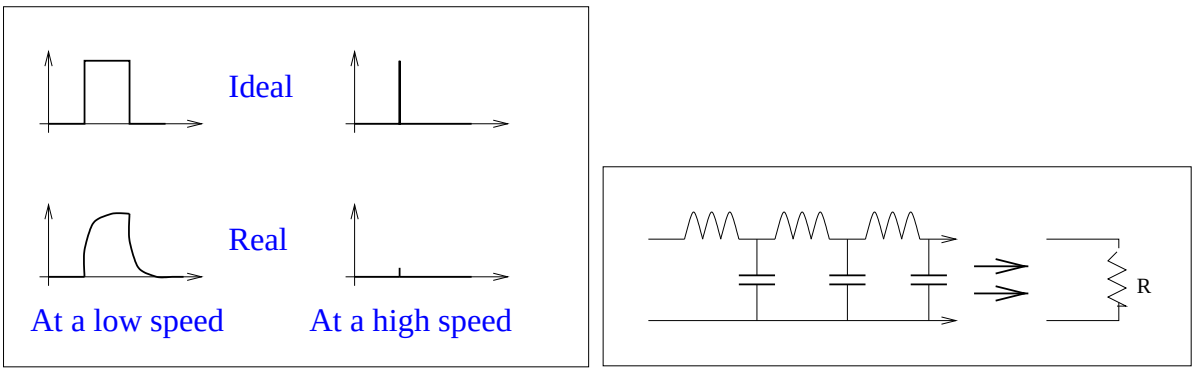


Figure 3.4: 'Real' signals on a cable.

We normally choose a high quality cable to get the best frequency characteristics, and set a maximum length (for example - with ethernet, 185m over RG58). The signals also reduce in level with distance. Over the 185m of an RG58 ethernet cable, the voltage level of the signal can drop to $\frac{2}{3}$ of its normal level. The speed of transmission in a cable is about $\frac{2}{3}C$ (200,000,000m/S).

However there is another cause of signal degradation, which is often more important. When our media is incorrectly *terminated*, we may get reflections of our signal. These reflections may *exaggerate* or *hide* our signal. To understand why this happens, we need to look a bit more closely at the characteristics of our media. We will do this by looking at a signal cable.

First - two definitions:

RESISTANCE the *quality* of a circuit which impedes the flow of *direct* current

IMPEDANCE the *quality* of a circuit which impedes the flow of *alternating* current

- A capacitor *will not* pass a direct current. Hence it has an *infinite* resistance.
- However it *will* pass an alternating current. A capacitor is said to have an *impedance* Z.

Cable model:

We can model a cable as an electrical circuit. In figure 3.4, we see one possible modelling of an electrical cable. We can model a *perfect* cable as a series of short segments, each containing a series *inductor*, and a *capacitance*. If the capacitance of our *perfect* cable over a length dx was Cdx, and the inductance was Ldx, the characteristic impedance is:

$$Z_c = \sqrt{\frac{L}{C}}\,\Omega$$

Note that this has dimensions of pure resistance[7], and so if we could construct an infinitely long cable its impedance Z (which varies with frequency) would tend toward a fixed resistance. Another way of saying this:

- if you wanted an infinitely long cable a resistor will model it.

This resistive value is called the *characteristic impedance* of the cable, and is dependent on the cable size, shape and material:

Line	Z_c
Twisted Pair	120Ω
Coaxial cable	50Ω→ 120Ω
Wire over ground	120Ω
Microstrip (In ICs)	120Ω
Parallel Wires	300Ω

If we put a step waveform, from 0 to V volts at time t_0 at one end of an infinitely long cable of characteristic impedance Z_c, the *incident* fronts of voltage (V_i) and current ($I_i = \frac{V_i}{Z_c}$) will move along the cable together at a speed of about $\frac{2}{3}C$. If instead of this infinitely long cable, we had finite cable with a termination resistor ($R_c = Z_c$), this will appear as an infinitely long cable attached to the end of our cable. When the voltage and current fronts reach the termination, the voltage across the line at any point is V_i, and the current in the terminator is $I_i = \frac{V_i}{R_c}$. We have a steady, stable state.

If however the termination resistor had a value $R_e \neq Z_c$, when the fronts reach the terminator, the current must be both $\frac{V_i}{R_c}$, and $\frac{V_i}{R_e}$. This is impossible since $R_e \neq Z_c$. We resolve this by noting that when the front reaches the terminator, a reflected voltage (V_r) and current (I_r) front begins travelling back down the line. The amplitude and polarity of the reflected front conserves the energy in the original step, and following the relation $\frac{V_i+V_r}{I_i+I_r} = R_e$.

What this means, is that we get a reflected signal when an incident pulse reaches a discontinuity in the impedance of a cable. We normally place a *resistor* with a value Z_c at the end of the cable to inhibit these reflections. Discontinuities in the impedance of the line can be either side of the correct impedance, and our reflections can be as large as the incident signal.

- If we *remove* the termination resistor, we get positive reflections.
- If we *short out* the remote end of a cable, we get negative reflections.

We can see from this that the correct termination of any media is vital to correct operation, and that connectors must be chosen to minimise reflections.

[7] You may wish to check this with dimensional analysis.

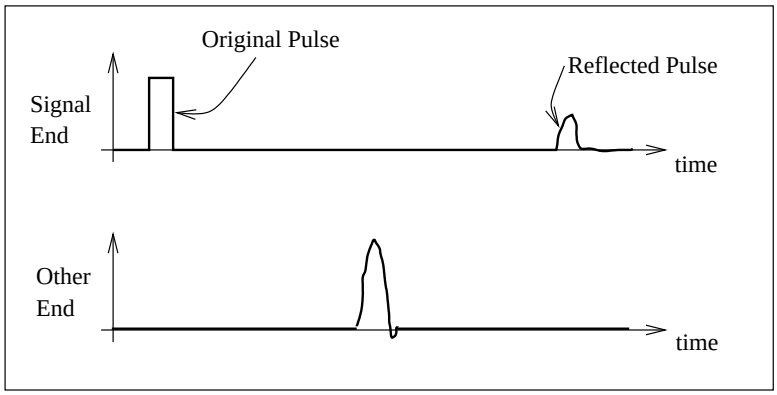


Figure 3.5: Reflections.

Summary:

Every terminator, T junction, cable bend (and so on) introduce deviations in the Z of the cable. These variations in Z cause positive or negative reflections, which may accumulate, and stop reliable operation of the cable. With 10 Mbps ethernet, we have 0.1 us pulses (one pulse in every 20 m), and reflections can change the levels of other pulses on the cable.

RG11 is a much higher quality cable (and hence lower noise). In addition it uses a method for tapping the cable which causes very little Z variation.

In either case, we must correctly terminate all networking cables with a resistor with a resistance $R = Z_c$ - the characteristic impedance of the cable.

3.5 Noise

It is normal to have up to 1 volt of random noise at some locations, rising to 100s of volts of noise in a fault. During a storm, it is common to have 1,000s of volts between buildings during lightning strikes.

We have to be careful with earthing to reduce noise. If you earth a cable at two different points, differences in the two earths may be significant.

In figure 3.6, we see two noise generators. With a single noise generator, even if the level of noise is much greater than the signal, the differential amplifier cancels out the signal, because it appears in both legs. However with two noise generators, the differential amplifier will amplify the *difference* between the two signals.

For this reason, earthing an *ethernet* cable in more than one place results in an *increase* rather than a *decrease* in noise.

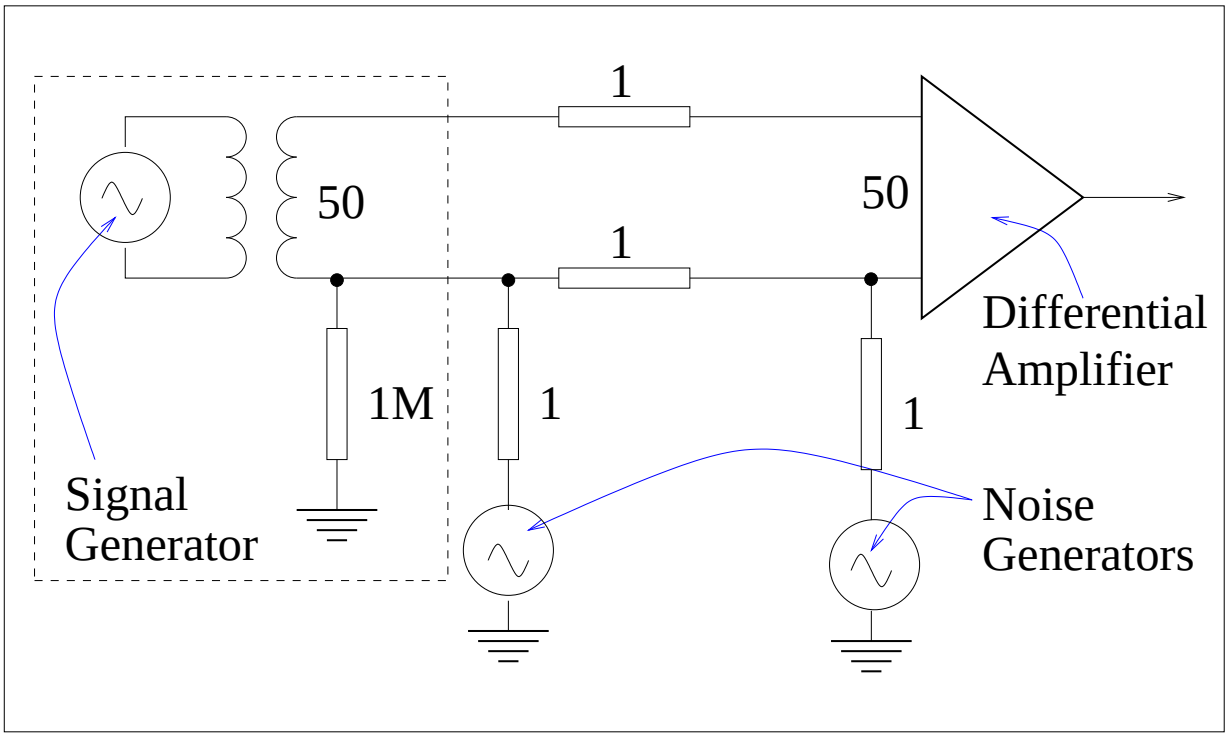


Figure 3.6: Noise generation.

Summary:

- Ethernet signalling is differential in order to cancel out noise common to both conductors.
- Applying an earth to both ends of a cable is like adding noise to one leg.
- Earth points are not perfect.
- Earth a differential cable in (at most) one place.
- Check the *whole* cable.

3.6 Electrical safety

When we interconnect machines, we must check that we are not being electrically unsafe. If we take ethernet as an example, all signals to and from an ethernet card are passed through electrical isolation circuitry for the following two reasons:

1. To stop *charge buildup* on the cable
2. For electrical *safety*

Each ethernet card has a 1 $M\Omega$ resistor between the signal ground and local ground. This does not provide electrical safety (A 1 $M\Omega$ resistor cannot *sink* much current). The function of the resistor is to stop the line building up a charge which might damage electronic circuitry, and perhaps give you a little shock.

However, some cabling systems (particularly thin ethernet) can be hazardous if misused. The ethernet cable has metallic connectors, which connect to the shield of the cable. If the cable is used to connect a machine in one building to machines in another building, and the cable is earthed in the other building (accidentally or otherwise), the cable may become an earth return for an electrical fault current.

For example, if there was an earthing fault in the other building, and 300V was placed on the cable shield, there are several scenarios:

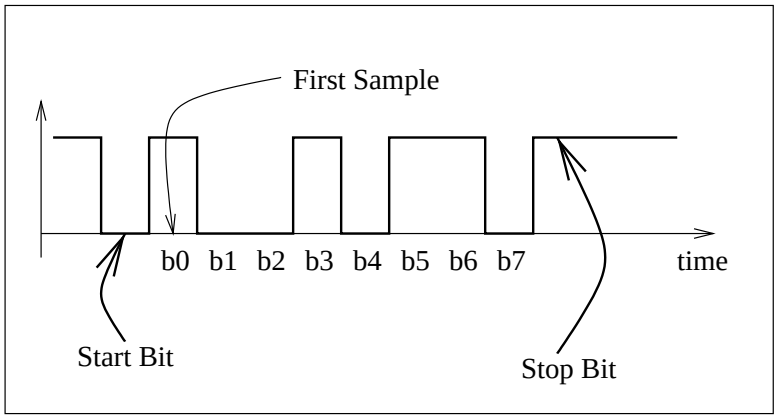
- You have also earthed the cable in your building - cable heats up, melts...
- You touch the cable while reaching round the back of the machine - you heat up, melt...

Summary:

- Ethernet should not be used to interconnect buildings without safeguards.
- Ensure systems are earthed correctly

3.7 Synchronization

We may send our data *synchronised* (meaning clocked) or *asynchronously* (without a clock):



Here we have asynchronous transmission. The reception algorithm is:

- Receiver listens for start bit transition

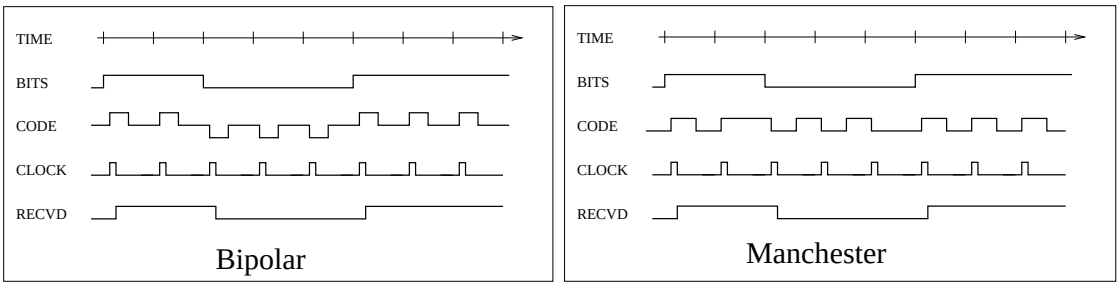
- waits 3T/2 to get b0
- then T to get b1
- then T to get b2 ... and so on ...

The implication of this is that both ends must agree on a rate of transmission.

Common asynchronous systems are found in the RS232 connection in the back of a PC. It is normally settable to data rates such as 300, 1200, 2400, 4840, 9600, 19200, 38400 bps.

Note: With RS232, we send a 0 as +12V and a 1 as -12V. This is called baseband in that we are not transmitting our bits by modulating another signal (as is done with a modem).

3.8 Digital encoding



In Bipolar encoding, a ’1’ is transmitted with a positive pulse, a ’0’ with a negative pulse. Since each bit contains an initial transition away from zero volts, a simple circuit can extract this clock signal. This is sometimes called ’return to zero’ encoding.

In Manchester (phase) encoding, there is a transition in the center of each bit cell. A binary ’0’ causes a high to low transition, a binary ’1’ is a low to high transition. The clock retrieval circuitry is slightly more complex than before.

3.9 Modems

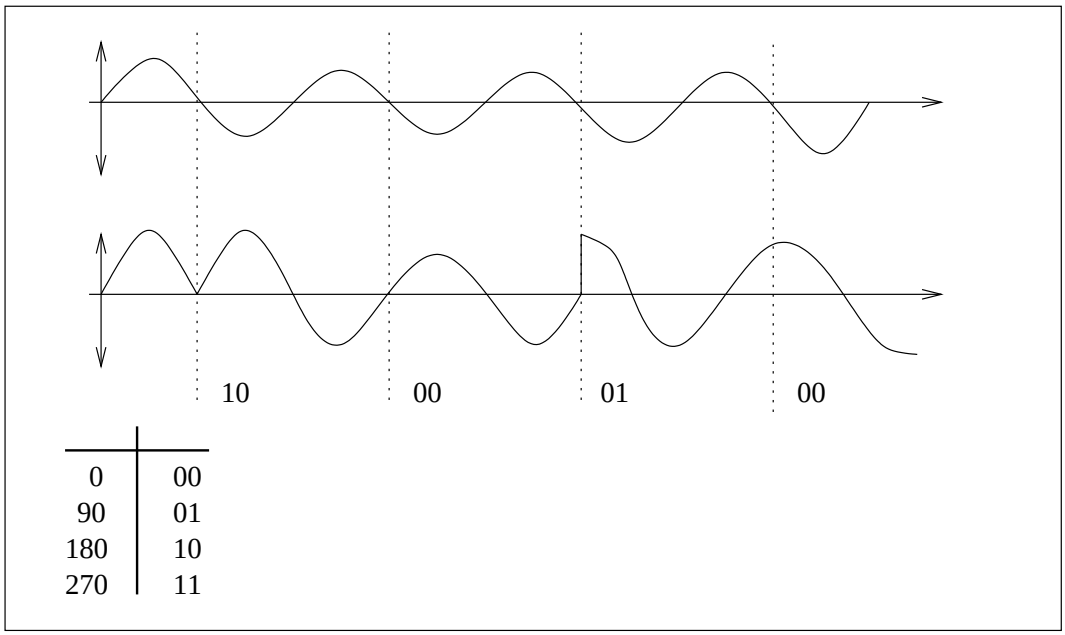
When we transmit data over a media which does not support one of these simple encoding schemes, we may have to modulate a carrier signal, which can be carried over the media we are using. The telephone network supports only a limited range of frequencies.

We use a range of modulation methods, often in combination:

- **AM** Amplitude modulation

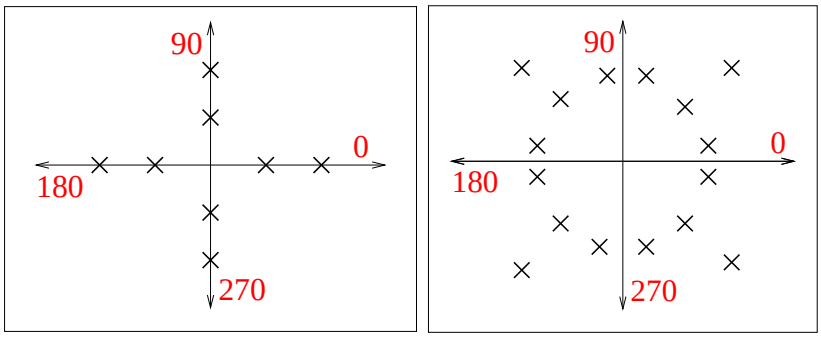
- **FM/FSK** Frequency modulation - frequency shift keying
- **PM/PSK** Phase modulation - phase shift keying

With a bandwidth of only 3,000Hz Nyquist shows us that there is no point in sampling more than 6,000 samples per second and if we only sent 1 bit per change in signal, we could only send 6,000 bits/sec. Common modulation methods focus on sending multiple bits per change to increase data rates (up to the maximum determined by noise - Shannon). The most common method is phase modulation, shown below:



0	00
90	01
180	10
270	11

We can also send different amplitudes at the different phases. The following phase plots indicate useful phase/amplitude values:



These schemes use multiple amplitudes and phases. They are called QAM[8]. The one on the left

[8]Quadrature Amplitude Modulation.

has 2 amplitudes and 4 phases giving a total of 3 bits per change. In the other example, we are sending 4 bits/change (4 bits/baud).

Common **modem** standards are:

- **V.32 bis** (14k4 => 14.4k) - 6 bits/baud => 64 points in the constellation.
- **V.34 bis** (28k8 => 28.8k) - 7 bits/baud => 128 point in the constellation.

Since modems are now quite complicated (they have computers, and can make all sorts of decisions to improve the transmission of data), we often need to communicate to the on-board *modem* computer. One common standard is that promulgated by the Hayes company. Many modems will respond to Hayes commands:

Command	Meaning
+++	Enter command mode
at&f	Reset to original settings
atdt2866	Dial number 2866
ath0	Hang up phone
at...	(and so on...)

Most modems do some or all of the following to reduce errors and improve speed.

- Add a parity bit to each 8 bits.
- Carefully choose where to place bit patterns in the constellation to reduce errors.
- use (software) compression of the data (MNP5).

3.10 Diagnostic tools

Many network faults are found at the physical layer, but there is no one tool to test for every possible fault. We can select from the following.

1. **Eye/Brain** - Most physical layer faults are visible or you can deduce what *must* be causing the fault.
2. **Multimeter** - A multimeter may be used for a cursory check, but it *can* pass a cable that should *fail*.
3. **TDR**[9]- The TDR emits a short pulse onto the line to be tested, and then listens for an echo.

4. **Replacement** - If you think a segment is faulty, replace it with a known good one. Not **another** one. A **good** one.

The TDR can check cables in ways that other tools cannot. If you have what you assume to be a good cable, a pulse put onto that cable should not echo. We have the following possibilities:

- **No Echo** -> cable is o.k. and terminated correctly.
- **Same polarity echo** -> cable has a high impedance mismatch (open circuit - or just bad).
- **Inverted echo** -> cable has low impedance mismatch (closed circuit - or just bad).

Note: Some TDRs will show you the reflected wave form, some just state good or bad.

The TDR can also measure the time between the pulse and the echo. We remove the terminator on the far end of the cable, and measure the time between the transmitted pulse and the reflected pulse.

- Distance to fault = $\frac{v*t}{2}$ where v is the velocity of the impulse.

Note: To get 1 meter resolution on a TDR, the pulse must be very short (typically 5nS). In order for the 5nS pulse to be safe to electronic circuitry, the amplitude should be less than 10 V. A 10V 5nS pulse does not travel very far along a cable before the group delay effect degrades it (200 m Max). To use TDRs on longer cables, you use longer pulses with a reduction in resolution.

[9]Time Domain Reflectometer.

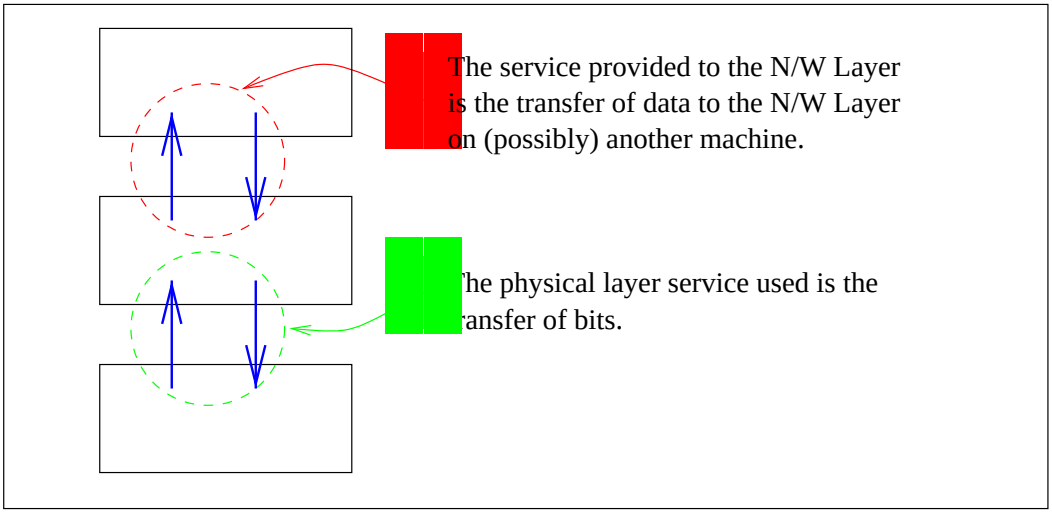
Chapter 4

Layer 2 - Datalink

Definition:

> *The datalink layer is concerned with the error-free transmission of data between machines on the same network.*

This layer normally constructs frames, checks for errors and retransmits on error. The datalink layer can be examined in terms of its interfaces - the service it provides to the layers either side of it:



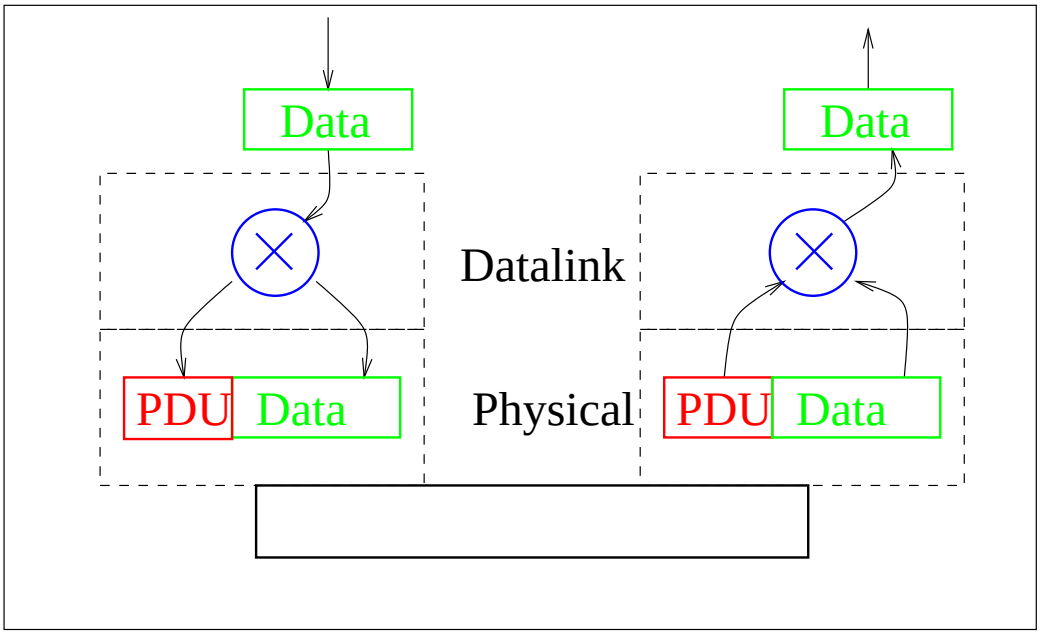
The *service* provided to the *network* layer may be:

- connectionless or connection oriented
- acknowledged or not acknowledged

Note: A Hub, transceiver or extender operate at the physical level. Bridges and etherswitches examine the datalink frame and so we say they operate at the datalink level. Routers operate at the network layer.

In order to provide the service to the network layer, the datalink layer may:

- frame the data
- deal with errors
- control the flow of data



The datalink layer on one machine communicates with a matching datalink layer on another machine, and must attach extra information to the transmitted data. In the figure, we see extra information being attached to data by the datalink layer software. This extra data is called a PDU[1].

The PDU might contain:

- machine addressing information.
- information to assist in error recovery.
- information to assist in flow control.

This technique of adding information to each outgoing message is repeated at each layer, so as the message moves down through the layers, it gets larger. Each layer prepends a PDU for the same layer on the other machine.

[1] Protocol Data Unit.

4.1 Sample standards

Name	Type	Frame size	S/Win.	Numbers	Error	Frame	Addressing
HDLC	P-P	arbitrary	Y	Y (1-8/127)	CRC-CCITT	Flag	1 byte
Ethernet	M/drop	1500	N	N	32 bit hash	Preamble	6 bytes
PPP	P-P	arbitrary	N	N	CRC-16/32	Flag	1 byte
LLAP	M/drop	600	N	N	CRC-CCITT	Flag	1 byte

In the table, we summarize some datalink layer standards, identifying areas in which they differ:

- Point to point or multidrop?
- Size of transmitted frame?
- Sliding windows?
- Are the frames numbered?
- Error detection scheme used?
- How are frames delineated?
- What size addressing?

4.1.1 HDLC

HDLC is derived from the original IBM standard SDLC[2], and is commonly found in use at sites with *mainframes*. A derived protocol LAPB[3], is an ISO layer 2 standard.

4.1.2 Ethernet

Ethernet is the term for the protocol described by ISO standard 8802.3. It is in common use for networking computers, principally because of its speed and low cost. In section 4.9, we examine the protocol used to resolve contention on an ethernet bus. Ethernet systems use *Manchester* encoded (see section 3.8), baseband signals.

4.1.3 PPP

PPP[4] is a protocol specially designed for interconnecting two computers, particularly when one *dials up* the other. It is protocol independent - that is, we may use PPP to transport IP, or IPX, or any network layer protocol. PPP includes a protocol to assist in setting up higher layer communication.

[2]Synchronous Data Link Control.
[3]Link Access Procedure B.
[4]Point to Point Protocol.

4.1.4 LLAP

LLAP is the Localtalk Link Access Protocol found on every Macintosh. It has some interesting qualities. For example:

- Node IDs for each machine are dynamically assigned.
- CSMA/CA[5]is used to reduce collisions.

4.2 Addressing

There are three main types of addresses needed at any level of the reference model:

1. **Machine** (or *source*, or *destination*) addresses.
2. A **broadcast** address (for messages to *all* machines).
3. **Multicast** addresses (for messages to *groups* of machines).

The first two are found on all systems, but the third may not be.

Ethernet

Each ethernet card is preprogrammed with a specific ethernet address. The addresses are six bytes, and are normally written as a series of six bytes separated by colons or full stops.

00:00:0c:00:42:b1

Each manufacturer of ethernet cards has a license to produce cards starting with a particular prefix, so you can tell who manufactured an ethernet card remotely, if you can find out its ethernet address. The ethernet broadcast address is ff:ff:ff:ff:ff:ff.

Ethernet has no defined multicast addresses, but it is possible to use any unused broadcast address. Win95 machines use the ethernet address 03:00:00:00:00:01 to communicate with each other.

Macintosh LLAP

The LLAP uses single byte addresses. Addresses in the range 1 to 127 are assigned to client machines, those in the range 128 to 254 are for server machines. The address 0 is unused, and the address 255 is a broadcast address.

[5]Carrier Sense, Multiple Access, with Collision Avoidance.

4.3 Modes

When we communicate between two machines, we classify the communication method into the following broad areas:

- **Half duplex** - each system transmits alternately (polite conversation)
- **Full duplex** - systems transmit at same time (New Zealand conversation)
- **Simplex** - one way transmission (Politician's conversation)

We also differentiate between:

- **Connection oriented**, and
- **Connectionless** protocols.

Name	Method	Advantages	Disadvantages
Connection oriented	set up link transfer data tear down link	secure ordered	slow
Connectionless	transfer data	no overheads fast	loss of data

CCITT use *connection oriented* protocols just about everywhere resulting in significant overheads.

4.4 Framing

How can we frame data?

We have three possible methods:

1. Put a count in the data. This scheme is very prone to error - if you miss the 'count byte', you may misinterpret the received data. It is seldom used.
2. Use physical layer violations at beginning and end of data. We have already seen how I^2C uses special start and stop conditions by violating the normal operational rules for I^2C data (see section 1.5.8).
3. Use bit or byte stuffing to differentiate between control and data signals. (The most common method).

4.4.1 Bit stuffing

With bit or byte stuffing, we specify an illegal bit or byte pattern in our data. If this pattern is received, it is outside of the frame. In bit stuffing, we use six 1s in a row:

> **Q:** What if the data includes five (or more) *1*s in a row?
>
> **A:** We stuff in an extra bit after every fifth *1*.

At the receiving end if we receive five *1*s followed by a *0*, we remove the *0*. This is done by hardware and is a standard framing method.

4.4.2 Byte stuffing

On some equipment (particularly Burroughs/Unisys), byte stuffing is used. A special character *STX* (in the ASCII table) identifies the start of text, *ETX* the end. If you want to transmit an *STX* or *ETX* you precede it with *DLE*. The receiver looks for *DLEs* and removes them, accepting the next character as a data value (whatever it is).

4.5 Error detection

It is possible to use ad-hoc methods to generate check sums over data, but it is probably best to use standard systems, with guaranteed and well understood properties, such as the CRC[6].

The CRC is commonly used to *detect* errors. The CRC systems treat the stream of transmitted bits as a representation of a polynomial with coefficients of 1:

$$10110 = x^4 + x^2 + x^1 = F(x)$$

Checksum bits are added to ensure that the final composite stream of bits is divisible by some other polynomial $g(x)$. We can transform any stream $F(x)$ into a stream $T(x)$ which is divisible by $g(x)$. If there are errors in $T(x)$, they take the form of a difference bit string $E(x)$ and the final received bits are $T(x) + E(x)$.

When the receiver gets a correct stream, it divides it by $g(x)$ and gets no remainder. The question is: *How likely is that $T(x) + E(x)$ will also divide with no remainder?*

> **Single bits?** - No a single bit error means that $E(x)$ will have only one term (x^{1285} say). If the generator polynomial has $x^n + ... + 1$ it will never divide evenly.
>
> **Multiple bits?** - Various generator polynomials are used with different properties. Must have one factor of the polynomial being $x^1 + 1$, because this ensures all odd numbers of bit errors (1,3,5,7...).

[6] Cyclic Redundancy Code.

Some common generators:

- **CRC-12** - $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
- **CRC-16** - $x^{16} + x^{15} + x^2 + 1$
- **CRC-32** - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$
- **CRC-CCITT** - $x^{16} + x^{12} + x^5 + 1$

This seems a complicated way of doing something, but polynomial long division is easy when all the coefficients are 1. Assume we have a generator $g(x)$ of $x^5 + x^2 + 1$ (100101) and the stream $F(x)$: 101101011.

Our final bit stream will be $101101011xxxxxx$. We divide $F(x)$ by $g(x)$, and the remainder is appended to $F(x)$ to give us $T(x)$:

```
          1010.010000
100101 )101101011.000000
        100101
          100001
          100101
             1001.00
              1001.01
                    10000
```

We append our remainder to the original string, giving $T(x) = 10110101101000$.

When this stream is received, it is divided but now will have no remainder if the stream is received without errors.

4.6 Error correction

There are various methods used to correct errors. An obvious and simple one is to just detect the error and then do nothing, assuming that higher layers will correct the error.

4.6.1 Hamming

The Hamming distance is a measure of how FAR apart two bit strings are. If we examine two bit strings, comparing each bit, the hamming distance is just the number of incorrect bits at the same location in the two bit strings.

If we had two bit strings X and Y representing two characters, and the *hamming* distance between any two codes was d, we could turn X into Y with d single bit errors. If we had an encoding scheme (for say ASCII characters) and the minimum *hamming* distance between any two codes was $d+1$, we could detect d single bit errors[7]. We can correct up to d single bit errors in an encoding scheme if the minimum *hamming* distance is $2d+1$.

If we now encode m bits using r extra *hamming* bits to make a total of $n = m + r$, we can count how many correct and incorrect hamming encodings we should have. With m bits we have 2^m unique messages - each with n illegal encodings, and:

$$\begin{array}{rcl} (n+1)2^m & \leq & 2^n \\ (m+r+1)2^m & \leq & 2^n \\ m+r+1 & \leq & 2^{n-m} \\ m+r+1 & \leq & 2^r \end{array} \text{ ““‘}$$

We solve this equation, and then choose R, the next integer larger than r.

Example: If we wanted to encode 8 bit values ($m = 8$) and be able to recognise single bit errors:

$$\begin{array}{rcl} 8+r+1 & \leq & 2^r \\ 9 & \leq & 2^r - r \\ r & \cong & 3.5 \\ R & = & 4 \end{array}$$

4.6.2 Feed forward error correction

In the previous example, each transmitted encoding depends only on the data you wish to transmit. *Convolutional* codes allow the output bit sequence to depend on previous sequences of bits. The resultant bit sequence can be examined for the most likely output sequence, given an arbitrary number of errors.

This encoding technique is computationally *inexpensive*, and is commonly used in modems.

4.7 Datalink protocols

These protocols can be simple - for example, if we do not care if the other machine receives the frame, we can just send it. This is what is done in ethernet.

```
Transmit(line,data);
```

[7] Because the code d bits away from a correct code is not in the encoding.

A slightly more sophisticated protocol might be one where the transmitter wants a positive acknowledgment that the frame has been received.

```
Transmit(line,data);
while not Receive(line)=ACK do
     {nothing};
```

The previous algorithm fails completely if no ACK is received. The general solution for this sort of problem is to introduce *timeouts* into our protocols to handle corruption or loss of messages.

```
repeat
   Transmit(line,data);
   SetSignal(timeout);
   while (Receive(line)<>ACK) AND not timedout do
       {nothing};
   if LastReceivedData=ACK then
      ReceivedOK := TRUE
until ReceivedOK;
```

A closer examination of this code indicates that it will also fail quickly under some circumstances[8], and it is clear that we have to code this carefully[9].

The three army problem also demonstrates that there are some simple situations with no deterministic solution. This particular problem is encountered when one or other computers using a connection oriented protocol attempt to shutdown or disconnect. We use protocols that give a good likelihood of success, or synchronize our machines in these circumstances.

If the transit time for a message is long, simple mesg-ack protocols can be unusable. If our RTT[10] is large, our data throughput can be reduced dramatically.

[8] You might want to examine the situation when an ACK for a previous message arrives late.

[9] In class we will examine six protocols in more detail - graded from simple ones like the first given above, right through to *safe* ones.

[10] Round Trip Time - If our messages were sent via satellite, we may have a large RTT.

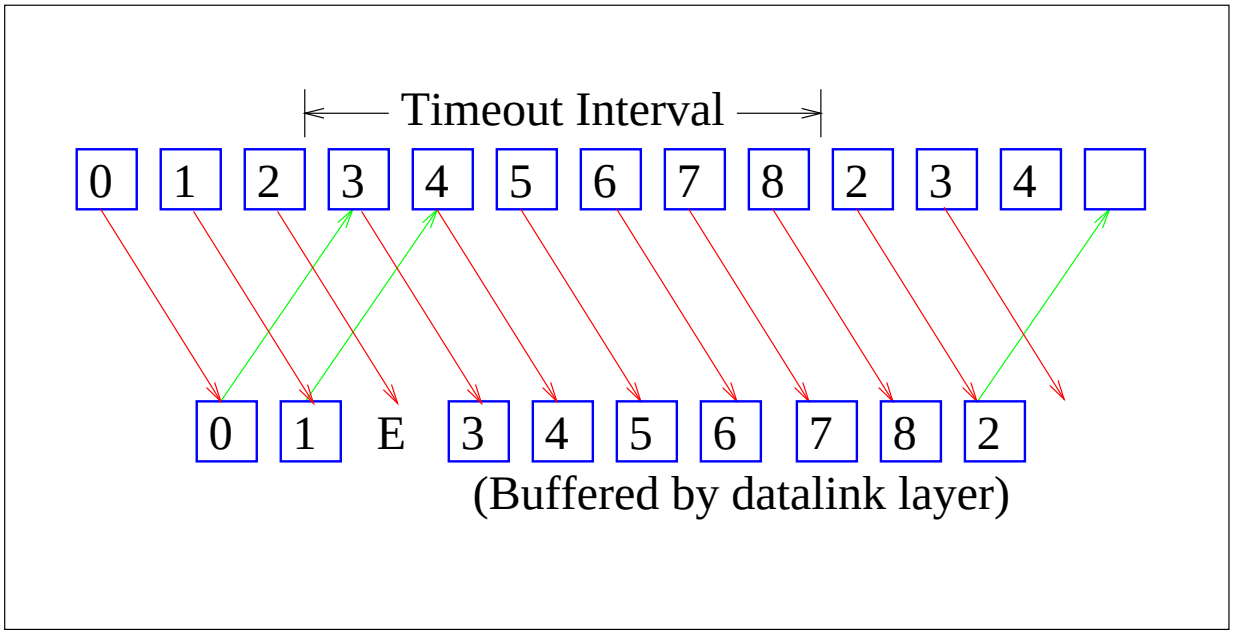


Figure 4.1: Sliding window protocols.

Example:

> With 1,000 byte messages sent via satellite (RTT=0.2sec) at 1 Megabyte/sec our speed is $\frac{1{,}000}{0.2} = 5,000$ bytes/second. This is obviously very slow and wasteful - it should take $\frac{1000}{0.002} = 500,000$ bytes/sec.

To solve this problem, we use sliding window protocols.

4.8 Sliding windows

When the ends of a transmission are remote from each other, the time for an acknowledgment that a frame has been received can become significant. In satellite transmissions for example, the 'bounce' time can be quite long. As well, some networks may store and forward data. For this reason, 'sliding window' protocols are used. These protocols attempt to handle any sequence of garbled, lost and out-of-sequence frames.

The method involves each frame containing a sequence number, and both the sending and receiving devices keep track of these numbers. The transmitter keeps track of all the unacknowledged frames in its sending window for retransmission if needed. Both the sender and the receiver require buffers to store the (unacknowledged) frames.

When the window size is 1, the method reverts to a simple 'msg-ack' scheme.

There are two flavours of sliding window protocols:

Go-back-N: The receiver stops acknowledging messages if one is lost. When the timer for frame #3 times out, since an ACK has not been received for it, the transmitter has to resend from frame #3. We have to buffer old transmitted messages. The size of this buffer is the *transmitter buffer size.*

Selective Repeat: The receiver acknowledges all frames it receives and also ACKs for ones it did not get. It can ACK just by not bothering to acknowledge the message. Note: We still have transmit window buffers and timers for each buffer. We also now have receive window buffers.

Note: Selective repeat is used in TCP.

Piggybacks

Often messages are going both ways at the same time. Messages can take along an acknowledgment for some previously received message. HDLC (high level data link control) uses these *piggybacked* ACKs. There are fields in the frame for both the message number, and the last received message number

TWIN = RWN = 8 buffers

4.9 MAC sublayer

We can either have point to point connections between machines or a shared (bus) connection. On a bus system we have two main problems

1. how to get (controlling) access to the media
2. what to do if there is a collision[11].

The MAC (Media Access Control) *sublayer* of the datalink layer handles this. ALOHA protocols[12] are commonly used.

- **Simple ALOHA** - Listen and then transmit if free
- **Slotted ALOHA** - Wait for slot, Listen and then transmit if free

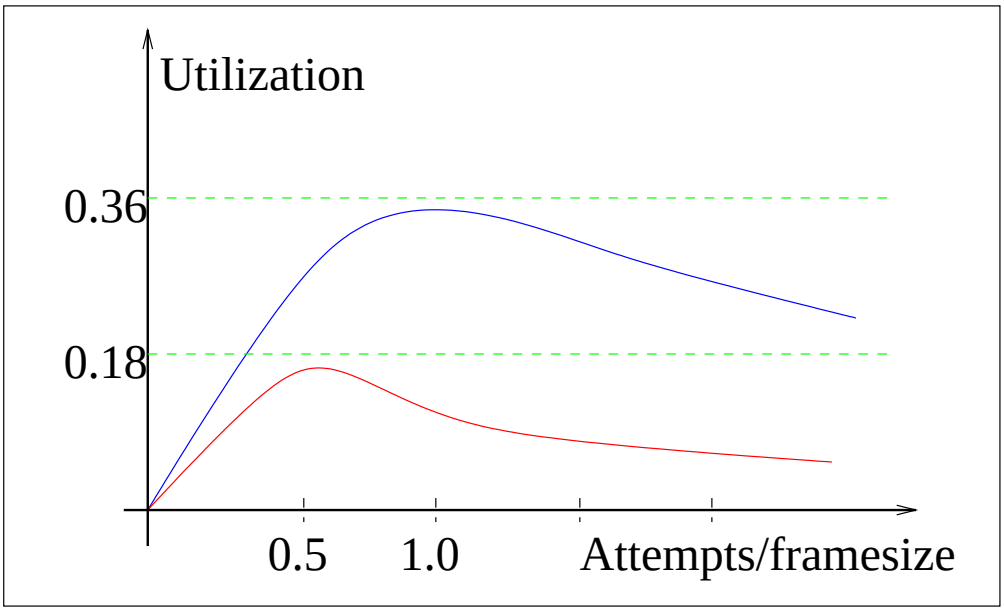


Figure 4.2: Utilization of links using ALOHA protocols.

In figure 4.2, we see the relative efficiency of links using both simple and slotted aloha. The utilization does not climb over $\frac{1}{e}$, but this does not mean that we can only have this limited utilization of the media. These values represent the situation when all nodes on a shared bus system are all attempting to transmit with equal probability.

In an ethernet system[13], machines on a smaller network can utilize close to 100% of the bus bandwidth.

4.9.1 CSMA/CD

Ethernet uses a system commonly called CSMA/CD. If a station wishes to send, it first checks the shared line for a carrier. Ethernet (manchester encoded) signals are added to a -1V DC signal. If the signal on the line is about zero volts, then the station can assume that the shared line is free, and can transmit:

```
while CarrierSense(line) do;   { Wait till line is free }
Transmit(line,data);           { Transmit the data }
```

[11]There are also various collision free protocols. These protocols generally use a 'contention' phase of operation in which one transmitting device acquires the channel. An example of this is the I^2C interface, which ensures that the device with the highest number succeeds. This contention phase is of course an overhead, but device addresses often have to be transmitted anyway.

[12]The ALOHA protocols were originally developed at the University of Hawaii (hence the name) to allow the distant parts of the campus network to inter-operate. The links were radio transceivers, all operating at a single frequency.

[13]Ethernet uses slotted aloha.

Note: In a system with two machines separated by 500m of RG11 cable, there may be a time difference of 2.5 uS.

The previous simple algorithm may fail to ensure that only one machine uses the shared (ethernet) bus at a time. Ethernet catches failures afterwards, using collision detection and binary exponential backoff (BEB) to handle failures.

Each station listens to what it transmits. If the message becomes garbled, it assumes that a collision has occurred, and then sends a burst of noise to ensure every station knows a collision has occurred. The station then waits for a time before retrying:

```
backoff := 1;
repeat
   while CarrierSense(line) do;   { Wait till line is free }
   if Transmit(line,data)=CollisionDetected then
      begin
         Transmit(line,noise);
         delay(random(backoff));
         backoff := 2*backoff;
         transmitOK := FALSE
      end
   else
      transmitOK := TRUE
until transmitOK OR (backoff>65536);
if backoff>65536 then
   { the transmission failed... }
else
   { it was OK! }
```

Note: There is a (small) possibility that a transmission will fail because 2 stations select the same random times to wait - 16 times in a row. This event is unlikely to occur on the USP network between now and the end of the universe.

4.10 Diagnostic tools

These tools are often platform dependent - for example on a P.C., if we were using packet drivers, there are a set of small diagnostic tools *(pktsend, pktchk*) which test the datalink software independently of other networking software installed on the computer.

Most networking cards come with card diagnostic software, but these will not test your *datalink* layers.

You may also find useful:

- **Arp** - indicates datalink and (IP) network addresses observed on a network.

- **Ping** - a network-layer aware echo request program (IP).
- **Ipxping** - a network-layer aware echo request program (for IPX).
- **Packetview, gobbler, packetman, packetmon** - tools to capture and display frames on your network.

Note: Tools that capture frames must put the ethernet card in a *promiscuous* mode. In this mode they will capture all frames - not just broadcasts and those addressed to your machine. This can cause lots of interrupts to your processor, and it is common for these tools to *lock up* the computers on which they run. In addition, some ethernet cards discard bad frames without telling you (i.e. no interrupts). This can lead you to think there are no bad frames on your ethernet.

With **packetman,** we can capture packets between any two machines, as long as one of them is on *our* network. The program shows:

Top Window - One line summary of all the packets. List of all the frames captured.

Center Window - Layer by layer breakdown of a selected frame (D/L, N/W, Transport, Application)

Lower Window - Hex byte - by-byte display of the selected frame.

Chapter 5

Layer 3 - Network

Definition:

> *The network layer handles routing and addressing for messages between machines that may not reside on the same local network.*

In an extended network, we are concerned with limiting the range of messages intended only for the local network, and routing messages that are to be delivered to a remote network. Network addresses are specified both for the source and the destination of the data. The OSI model allows for any addressing scheme, and specifies codes for all the common addressing formats[1]. The format specification part has plenty of room for expansion for future addressing schemes. The addresses may be of variable length.

To help identify the source and destination of messages, it is common to partition machine addresses into:

- a *host* part, and
- a *network* part[2].

This simplifies the task of our software responsible for delivering messages. The messages contain destination addresses (and normally source addresses), and it is easy for the software to determine if the message should be delivered to a machine on the local network, or sent to another network. The difficult question here is:

> *Which other network?*

There is no simple answer, and the routing software on an extended network may be complex.

[1] Formats for telephone numbers, ISDN numbers, telex numbers, and so on.

[2] Sometimes we reuse datalink addresses, but sometimes we use a whole new addressing scheme for these *network* addresses.

Service provided to the transport layer:

The network layer hides the network type and topology from the transport layer. It also provides a consistent addressing scheme. The OSI framework allows for *two* types of service:

- Connection oriented, and
- Connectionless (datagrams).

The OSI network model specifies the following service areas:

Connection and disconnection: Connection primitives provide for setting up a connection through the network. The disconnection primitives provide for the orderly termination of the connection.

Data transfer/expedited data: Data transfer primitives provide for ordered transfer of data through a previously set up connection. The expedited data transfer primitives allow for a data packet to be sent ahead, bypassing the normal ordering and queuing schemes.

Data transfer/unitdata: Unitdata is a connectionless data transfer - a *datagram*.

Reset: Reset primitives provide for recovery after detection of some failure in the system. All data is lost.

Report/status: Status services provide information on the status of the network.

Network interconnection:

The interconnection of networks (especially dissimilar ones) is a complex issue. Some networks cannot be connected together without losing some property of one of the networks.

For example: the Token Bus network has a 'fast acknowledgment feature. If a bridge acknowledges a received frame, and then that frame cannot be routed to the destination, the bridge has *'lied'*. If on the other hand the bridge does not acknowledge the frame, the sending unit will deduce that the destination is not available, which may also *not be true*.

5.1 Sample standards

In table 5.1, we summarize some network layer standards, identifying areas in which they differ:

Name	Addressing	OOB	Windows	Window size	Packet size	Numbering	Fragment
IP	4	Y	N	-	65,536	N	Y
IPv6	16	Y	N	-	65,536	N	Y
IPX	10	N	N	-	512	N	N
X.25	variable	Y	Y	8 or 128	128	Y	Y

Table 5.1: Sample network layer standards.

- Bytes for address?
- Out of band data?
- Sliding windows?
- Size of the window?
- Size of packets?
- Numbering scheme?
- Allow fragmentation?

IP and IPv6

IP[3] is perhaps the most widely distributed network layer protocol. It belongs to a set of protocols, called IP, developed over the last 25 years. Initially the IP suite was developed for the US military as a research project into fault[4] tolerant networks. The protocols cover from *network* to *application* layers, and are continually being developed.

IPv6 is a development of IP, giving a larger address space, and support for alternative carriers.

IP is documented in a set of documents called the RFCs[5]. There is an RFC for every protocol in the IP suite. An RFC is initiated by anyone who wishes to specify a new protocol and they are commented on/vetted/improved by the internet community before final distribution.

IPX

As a contrast, IPX[6] is a network layer protocol for use with Netware file servers. It is distributed by Novell Inc, and is a proprietary protocol. It is seldom used for anything except interaction with Netware file servers.

X.25

Telecom provide *'connection oriented'* service with X.25. Many users put their own protocols *on top* of it. This may lead to inefficiencies - connection oriented services on top of connection oriented services.

[3] **I**nternet **P**rotocol
[4] The *fault* the US military was concerned with could tactfully be called a *'nuclear'* issue...
[5] Request for Comments.
[6] **I**nternetwork **P**acket e**X**change.

X.25 defines an interface between DTE and DCE devices on public data networks. It is a CCITT recommendation, and has been adopted by every country providing PSDN services. X.25 in no way defines the internal methods by which the PSDN routes and switches the service.

The facilities provided by X.25 include:

- Connection oriented data transfer
- Connectionless (datagram) data transfer
- Sliding windows up to 128
- Selection of different carriers
- Various charging options

5.2 Addressing

5.2.1 IP Addressing

The IP network layer addressing scheme uses four bytes[7], and is often written as dotted decimal, or hex numbers:

Decimal	Hex
156.59.209.1	9C.3B.D1.01

This address defines an interface not a host. - a host may have one, two or more interfaces, each with different IP network layer addresses[8].

- Each interface has at least one unique address.
- Machines on the same network have similar addresses.
- For every interface on an IP network, you have not only the *IP address* but also a *mask*, which defines the *network* part of the address.

[7]Four bytes will allow over 4,000,000,000 different machine addresses, and this was considered adequate 25 years ago. However a wasteful allocation scheme has resulted in much of this address space being used up.

[8]The IP *network* layer address is often called the *'IP address'*.

5.2.2 IP network masks

An IP network mask looks like an address, but the meaning of the bits are different. In a network mask, the bits identify the *network* and *host* parts of the address:

- If the bit is a 1, it is part of the *network* address.
- If the bit is a 0, it is part of the *host* address.
- The host addresses are normally consecutive, but *need not be*[9].

For example:

```
Interface address:  156.59.209.1  -> 10011100 00111011 11010001 00000001
Network Mask:      255.255.252.0 -> 11111111 11111111 11111100 00000000
(AND)
Host part:                                                01 00000001
Network part:                        10011100 00111011 11010000 00000000 -
> 156.59.208.0
```

The host is **156.59.209.1** on network **156.59.208.0**, with a network mask of **255.255.252.0**.

Machines can determine if they are on the same network by **AND**ing their host address with the network mask. If the resultant network addresses are the same, the two hosts are on the same network. Choosing an incorrect mask can result in confusion.

Two special addresses are reserved on any IP network:

1. The one with the host part all 0s.
2. The one with the host part all 1s.

You cannot allocate these addresses to machines. They are used for broadcasting to all machines on the same network.

5.2.3 IPX addressing

The IPX network layer protocol has:

- a *network* part (four bytes), and

[9] It is allowable to choose a mask with the *host* bits scattered throughout the *mask* bits. This would result in host addresses scattered over a range, *not* consecutive.

- a *host* part (six bytes)

This gives a total of ten bytes for the IPX address. It is written as follows:

93:3B:01:00:00:08:C0:31:55:24

From this address it is easy to determine the datalink address of the machine (00:08:C0:31:55:24) and the network on which it is found (93:3B:01:00).

5.2.4 Appletalk Addressing

We have already seen the 1 byte addressing scheme used in the *localtalk* datalink layer. The Apple network address is a 3 byte one, a 2 byte network part and 1 byte for the host. Once again, this addressing is hidden from the user, and dynamically set up[10].

Note: the Mac architecture imposes a low limit on the number of interconnected Macs on a network.

5.3 IP packet structure

Figure 5.1 shows the structure of an IP packet. The *'ihl'* field gives the header length in 32 bit chunks. Notice that our IP addresses fit into the 32 bit source and destination address fields.

The *'TTL'* field gives the *time to live* for the packet. Each time the packet passes through a router, it is decremented by 1. If *TTL* reaches zero, the router sends the packet back to the source address. This has two uses:

1. You won't get a packet looping forever.
2. You can test reachability by artificially setting *TTL*. (see the item on *traceroute* in section 5.8).

[10] You don't have to do anything. Macs configure themselves.

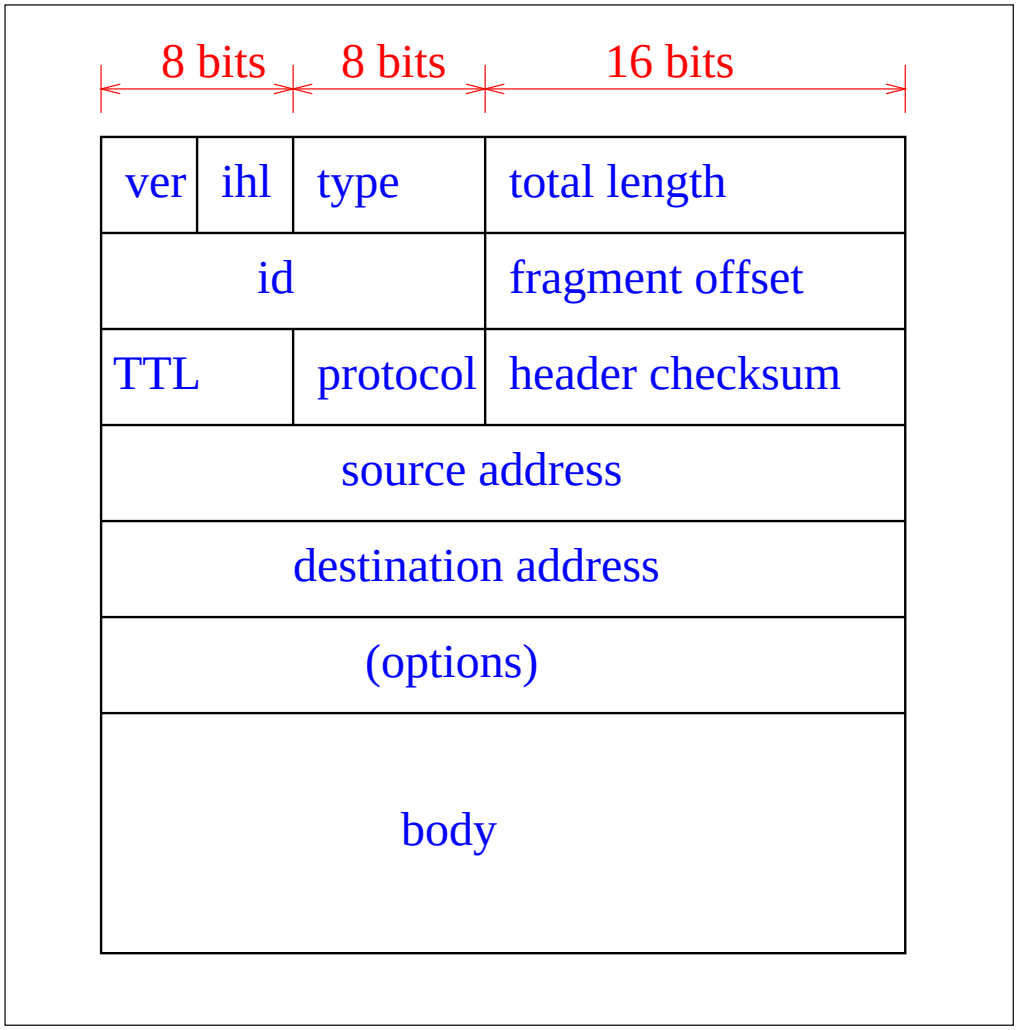


Figure 5.1: IP packet structure.

5.4 Allocation of IP Addresses

Worldwide, there are five classes of IP address ranges:

Class	Who for	Prefix	Size	Number
A	Huge organization	0xxxxx...	2^{24}	2^{7}
B	Large organization	10xxxx...	2^{16}	2^{14}
C	Small organization	110xxx...	2^{8}	2^{21}
D	Multicast	1110xx...	1	2^{28}
E	Reserved	11110x...	1	2^{27}

Note: there is no way to ask for a single organizational IP address, and this accounts for the unfortunate situation we are getting to - nearly all the IP addresses have been allocated!

There are some solutions to the IP address space problem:

1. Private IP addresses - accessable via a gateway.
2. IPv6 - has a 16 byte address space.
3. Reuse of addresses in remote regions, with router support.

5.5 Translating addresses

Computers on a network often need to translate to and from network and datalink addresses.

An example:

If a machine knows the network address of a machine it wishes to transmit to, it can find out if the other machine belongs to the same network. If it does, the machine needs to find out the datalink address of the other machine, so that it can directly send the message.

The protocol to do this is called ARP[11].

[11] Address Resolution Protocol.

ARP

An **ARP request** for the specified *network* address is sent to the *datalink* broadcast address. Machines that know the translation between the network and datalink address respond with an **ARP response**, containing the *datalink* address for the specified *network* interface.

Machines normally maintain ARP tables containing results of recent ARP requests. You may query these tables using the **arp** command:

```
opo 30% arp -a
manu.usp.ac.fj (144.120.8.10)  at 0:0:f8:5:6a:a1
kula.usp.ac.fj (144.120.8.11)  at aa:0:4:0:b:4
teri.usp.ac.fj (144.120.8.1)   at 0:0:f8:31:1c:da
?              (144.120.8.125) at aa:0:4:0:32:5
?              (144.120.8.251) at aa:0:4:0:7f:6
opo 31%
```

5.6 Routing

The particular routing scheme used by the network layer is normally hidden from the network layer *user*. However there are two main schemes used internally:

- Fixed routing computed at connect time
- Individual packet routing done dynamically

If you have a group of networks connected by routers, we have to distribute routing information. RIP is one such router protocol. Routers listen for, and broadcast RIP[12] packets out all interfaces. In this way, the routers can learn about adjacent networks. You can query the state of routing tables on most routers. The structure of IP subnetting minimizes the number of routes that a router has to keep track of. It is common for smaller machines to be given a default route (or gateway) rather than letting them sort it out using RIP.

There are other protocols for routing, such as IGRP - the Internet Gateway Routing Protocol.

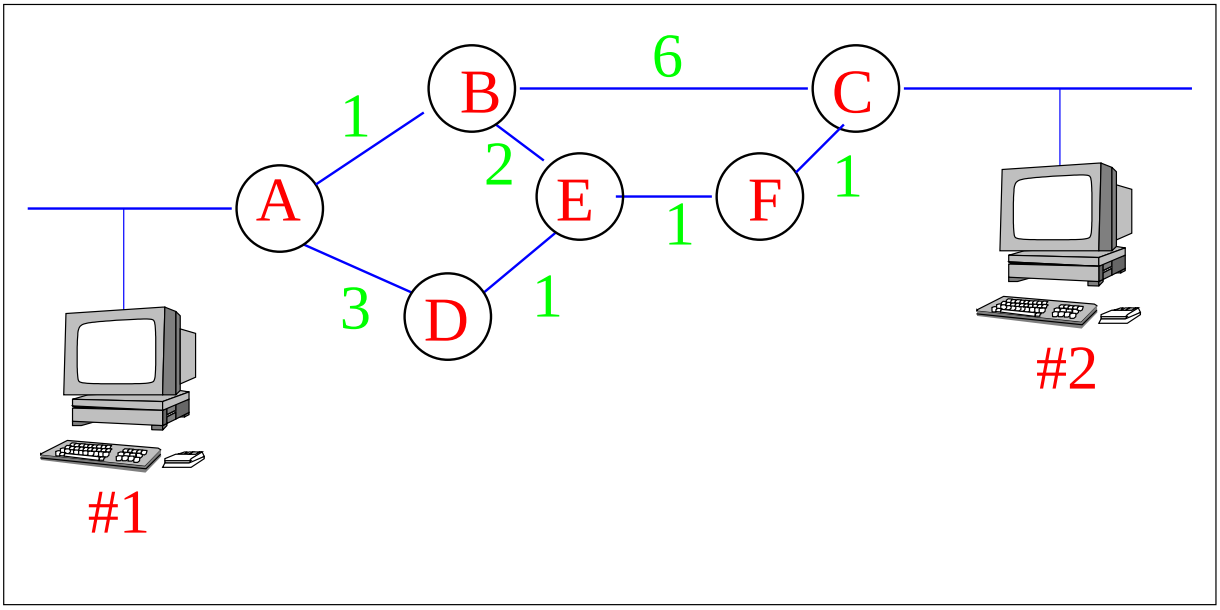
5.6.1 Routing Protocols

There are many routing algorithms, and we will just look at a few.

[12]Routing Information Protocol.

Static or fixed routing

We might use a technique such as the shortest path - here a path could be *how long* or *how many hops* or some other metric.



If **hops** were used for our metric, **ABC** is the shortest path from machine #1 to machine #2. However, if **delay** was used, and AB=1, AD=3, BE=2, DE=1, EF=1, BC=6 and FC=1, then the metric attached to **ABC** is 7 and **ABEFC** is only 5.

In the above simple example, we could use fixed tables, and preload each router from these tables.

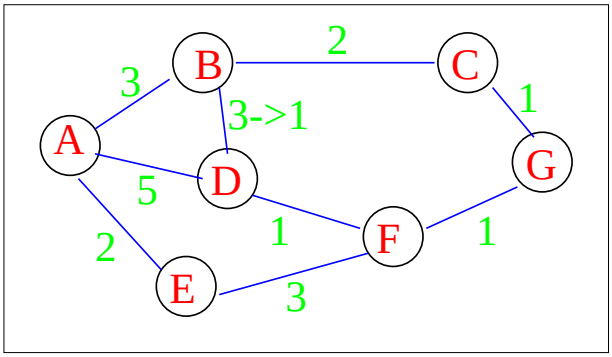
There are various algorithms involving *walks* through the network which calculate the shortest path through a network, but you should note that static routing will not respond to changes in the network.

Dynamic routing

One common method used to dynamically find effective routes is the *'distance vector'* scheme, used by RIP.

In distance vector routing, each router maintains a table of distances to all other routers indexed by router number. Periodically, attached routers exchange information about their tables.

In the following diagram, we have an internetwork with seven routers. The figures represent the metrics attached to each route, and note that the route BD is about to change from a metric of 3 (meaning not so good) to 1 (meaning good).



Before the change, router D has the following table, giving it's view of the best routes:

Destination	Delay	Route
A	5	A
B	3	B
C	3	F
D	0	-
E	4	F
F	1	F
G	2	F

Router B then sends to D it's version of the router table:

Destination	Delay	Route
A	3	A
B	0	-
C	2	C
D	1	D
E	5	A
F	2	D
G	3	C

Router D then rewrites it's table to reflect the new information:

Destination	Delay	Route
A	4	B
B	1	B
C	3	F
D	0	-
E	4	F
F	1	F
G	2	F

Router D can determine from router B's information that there is a better route to router A than the direct one.

Note: This algorithm responds to good news quickly and bad news slowly[13].

5.7 Configuration

5.7.1 Addressing

We have already seen how datalink addresses are configured or set:

- Ethernet addresses are set by having a PROM on the ethernet card.
- Token ring systems generally have an 8 bit switch for setting the token ring address.
- Local Talk uses a dynamic scheme, to make the Mac networks self configuring. When a Mac starts up, it chooses a datalink address, and then broadcasts to the chosen address. If a machine replies, the Mac chooses a different address, and tries again.

There are several strategies for setting network layer addresses:

- Manually
- Determine it from the datalink address (as used by IPX)
- Dynamically retrieve one from a server

Dynamic methods are considered best here, as we can centrally administer the addresses used, and then publish them to the machines, without having to go to each machine and 'set' it. Machines wishing to find out their IP information broadcast a query. A server responds with the requested information. Here are common protocols:

RARP - Reverse Address Resolution Protocol: The Amoeba processors in the Mixed lab use RARP to configure and boot themselves. RARP translates a *datalink* address to a *network* address.

BOOTP - BOOT Protocol: This protocol is intended to be used for diskless booting of machines. It can also be used for IP configuration information.

DHCP Dynamic Host Configuration Protocol: A newer version of BOOTP, with one major improvement. - it allows for *leasing* of IP addresses.

[13]The problem here is commonly known as the *count-to-infinity* problem. The routers slowly add to their metrics for the bad path - each router thinking it has a slightly better path through another misinformed router.

5.7.2 Routing

The configuration of routing information is normally best left to the protocols that are supposed to do it. However, sometimes you have to configure routing information.

IP machines

In Win95, you will have to set the default route in the IP properties window.

UNIX and NT machines are generally self-configuring, but you may use the *route* command to add static routes. For example: to add a route to network **212.232.32.0** via gateway **128.32.0.130**, use:

```
route add net 212.232.32.0 128.32.0.130
```

You may also use the netstat command to display RIP information for your local router:

```
netstat -r
```

IPX machines

Netware servers with multiple NICs can perform IPX (and IP) routing between the networks. In the netware configuration script executed at boot time, you bind protocols to each of the NICs, and if IPX is bound to both NICs, routing software in the server will route between them.

Note: In some of the Netware documentation, they incorrectly call this *bridging*!

Macs

Mac routers normally have both localtalk and ethertalk ports. Any configuration of the boxes is generally done over the network using Mac software.

Mac have their own terminology - they use the term *zone* instead of *network*. When Macs are connected to a network with routers in it, the Mac chooser has an area for selecting which *zone* you are interested in. The area has names (not numbers) in it. These names are set by the router. You can either let the router choose names and numbers, or you can specify them.

5.8 Diagnostic tools

Network layer diagnostic tools are generally based on the original UNIX IP based systems, and examination of the relevant UNIX commands will help in understanding[14].

You may find useful:

arp: Displays and controls ARP tables.

ping: Sends ICMP ECHO_REQUEST packets to network hosts. This command can be used to check for basic network connectivity:

```
manu> ping turing
PING turing.usp.ac.fj (144.120.8.250): 56 data bytes
64 bytes from 144.120.8.250: icmp_seq=0 ttl=64 time=1 ms
64 bytes from 144.120.8.250: icmp_seq=1 ttl=64 time=0 ms
----turing.usp.ac.fj PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip (ms) min/avg/max = 0/0/1 ms
manu>
```

netstat: Displays routing and network status and statistics:

```
opo 36% netstat -r
Destination Gateway        Netmask    Flags Refs    Use Interface
default     reba.usp.ac.fj             UGS      1  11012 ec0
144.120.8   opo.usp.ac.fj  0xffffffc00 U       10 423827 ec0
opo 37%
```

traceroute: Displays the route that packets take to the network host. This is done by setting the TTL value in an IP packet to 1. The first router then sends it back, giving you the transit time to the first router. The TTL is then set to 2, giving you the transit time to the second router, and so on:

```
manu> traceroute kai.ee.cit.ac.nz
traceroute to kai.ee.cit.ac.nz (156.59.209.1), 30 hops max, 40 byte pack-
ets
1 reba                (144.120.8.16)       2 ms   2 ms   3 ms
2 202.62.125.133      (202.62.125.133)   572 ms 579 ms 222 ms
3 202.62.120.1        (202.62.120.1)     400 ms 393 ms 399 ms
4 202.84.251.5        (202.84.251.5)     412 ms 595 ms 418 ms
5 s4-3a.tmh08.hkt.net (205.252.128.157)  825 ms 806 ms 773 ms
6 s4-3a.tmh08.hkt.net (205.252.128.157) 1141 ms 979 ms 786 ms
...
```

packetman: Other useful tools are packet capturing and analysis tools such as packetman.

[14] Use the *man* pages for each command.

Chapter 6

Layers 4,5 - Transport, Session

Definitions:

> *The transport layer ensures a network independent interface for the session layer. We can specify how secure we want our transmission to be in this layer. The transport layer isolates the upper layers from the technology, design and imperfections of the network.*
>
> *The session layer is closely tied to the transport layer (and often the software is merged together). The session layer is concerned with handling a session between two end processes. It will be able to begin and terminate a session, and provide clean 'break' facilities.*

At these higher layers of the reference model, we are dealing with software, and we may be involved with:

- Writing software to interact with the protocol stack, and
- Configuring the protocol stacks.

When we write software for these layers, we use standard APIs[1].

OSIRM

In the transport layer, the reference model provides for five levels of *'class of service'* ranging from a simple connection manager (class 0) to a full error and flow control manager (class 4). You would use class 0 when communicating over a PSDN, and class 4 when using a PSTN.

In the session layer, the reference model provides for:

[1] An API is the **A**pplication **P**rogrammer's **I**nterface.

Name	Addressing	Communication	Windows	Window size
TCP	IP address,port	session	Y	variable
UDP	IP address,port	datagram	N	-
NETBEUI	nodename	datagram & session	N	-
SPX	IPX address,port	session	Y	8 or 128

Table 6.1: Sample transport layer standards.

- Setting up sessions with another entity on another machine,
- Synchronizing sessions at agreed points, and
- Handling interrupts and exceptions to the normal flow of information.

IP

The ARM[2] has only four distinct layers[3]:

- **Network Interface** - This layer encapsulates all the hardware dependencies.
- **Internet** - This is similar to the OSIRM *network* layer, and has only one implementation - IP.
- **Host-Host** - This is similar to the OSIRM *transport* layer, and has various implementations.
- **Process** - This is similar to the OSIRM *application* layer, and has protocols for just about anything.

6.1 Sample transport standards

In table 6.1, we summarize some transport layer standards, identifying areas in which they differ:

- Addressing?
- Type of communication?
- Sliding windows?
- Size of the window?

[2]The **A**rpanet **R**eference **M**odel.

[3]The terms used are the ones from the ARM. ARM and OSIRM are distinct network architectures, though there is a rough correlation between some of the OSIRM layers and ARM ones. The ARM architecture is not a *cut-down* version of the OSIRM. It has a clear layered structure and has been useful for many years. The Internet is *built* on IP/ARM.

IP

The Internet protocol suite has two common *transport* protocols:

- **TCP** - Transmission Control Protocol. The extra information found in the PDU for TCP is: source, destination, sequencenumber, acknowledgenumber, windowsize and so on[4].
- **UDP** - User Datagram Protocol. The extra information found in the PDU for UDP is source, destination, length, checksum.

Netbeui

Netbeui[5] was developed by IBM in 1985, and was a protocol focussed on small LANs, segmented into small groups of computers. It is commonly used in WfW, LAN Manager, and NT.

In common with our other network and transport layer protocols, Netbeui can be found on any datalink type, and on many platforms.

SPX

SPX is the sequenced protocol used by Netware file servers. Originally, Netware used the network layer protocol IPX for file serving, but the demands of larger networks led to the introduction of a transport protocol.

6.2 Session standards and APIs

Most network programming involves using an API which provides a *session layer* service. The API provides a set of system calls, and software which implements some *view* or *abstraction* of the network.

It is normal to use these programming *abstractions* when doing network programming. They allow you to model the behaviour of the system, and understand its behaviour. Before programming using one of these APIs, you need to understand the abstract model, not the underlying protocols.

[4] This is exactly what you would expect from a *sliding window* protocol.

[5] **N**etwork **B**asic **E**xtended **U**ser **I**nterface.

Netbios

Netbios[6] was developed by IBM. It is an interface to low level networking software, and through the API, you can send and receive messages to and from other Netbios machines. The Netbios calls on a PC are through Interrupt 5c, and require your software to initialize and maintain an NCB[7] before calling the network software.

Sockets

The *'socket'* is an abstraction for each endpoint of a communication link. The abstraction allows data communication paths to be treated in the same way as UNIX files. When a socket is initialized, the API system calls return a *file descriptor* number which may be used to read and write in the same way as those returned by file opening calls.

Protocol families supported include Appletalk, DECnet and IP. In the IP world, the socket API primitives support both TCP and UDP communications. Sockets are often used when implementing client/server applications.

Remote Procedure Calls

The RPC system was introduced by Sun, but unfortunately there are variations in its implementation. The intent is to provide a system in which networking calls look and act just like procedure calls. The program prepares parameters for the procedure call, and then makes a call to a stub procedure. The stub procedure uses the RPC runtime software to transfer the parameters, only returning when the call is complete.

RPC is responsible for ensuring that the parameter data is transferred and returned correctly.

6.3 Addressing

We may have a different addressing scheme at each layer. At the network layer, the address refers to an interface to a machine. This is sometimes called the NSAP[8] address.

At the transport layer, we have TSAPs (**T**ransport **S**ervice **A**ccess **P**oints), typically an NSAP and a port number. This address identifies a particular data communication *end point*.

[6] A networking 'BIOS'. BIOS is the term used on IBM personal computers for the low level software that hides the underlying hardware in the system. It is often found in a PROM. The letters stand for **B**asic **I**nput **O**utput **S**ubsystem.

[7] **N**etwork **C**ontrol **B**lock.

[8] **N**etwork **S**ervice **A**ccess **P**oint.

Netbeui and Netbios

The NCBs contain an address constructed from a *logical session number*, and a *netbios name* (up to 15 characters).

SPX

SPX endpoints are identified by an IPX address, and a (16 bit) integer port number.

Sockets

When initializing a *socket*, we specify the '*address family*', the *'transport mode*' and the '*protocol*'. With a normal IP socket, this may lead to a socket endpoint with:

- a protocol: TCP/IP,
- an address: 156.59..., and
- a port: 2145

6.4 Transport layer

In the transport layer, we see again systems introduced in earlier chapters:

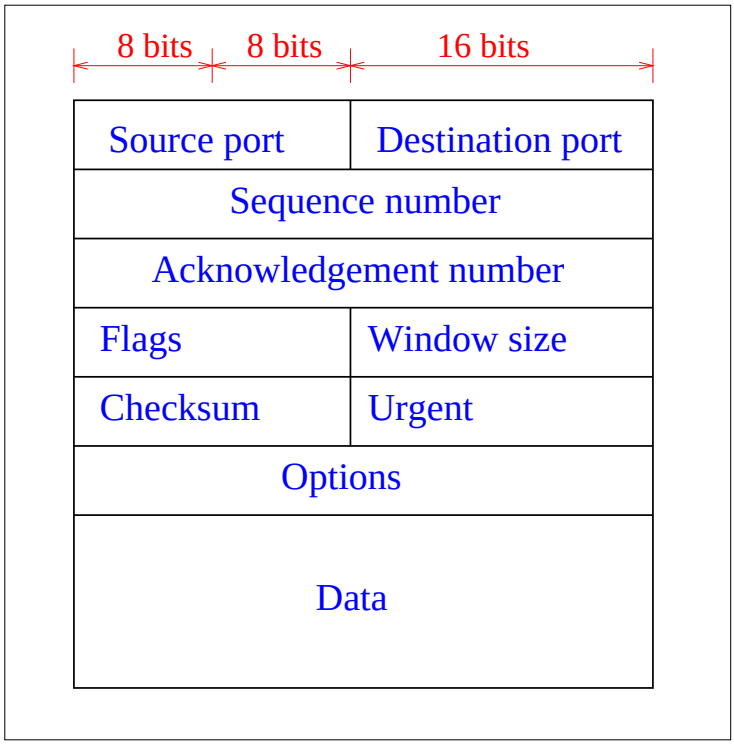
- Connectionless and connection oriented transfers
- Sliding windows
- Flow control
- Error recovery.

We will look at two of the IP protocols.

6.4.1 TCP

TCP is a connection oriented and secure protocol, designed to be *safe* in a wide range of environments. It can be used for slow bit rate point to point links via an undersea cable, or fast LAN use. It is defined in RFC793.

TCP software packetizes a data stream in chunks of less than 64kbytes and attaches extra sequencing, and target process (port) information to each packet. Here is the TSAP header:



From the diagram, we can see that TCP supports piggybacked acknowledgements in a conventional go-back-N sliding window protocol.

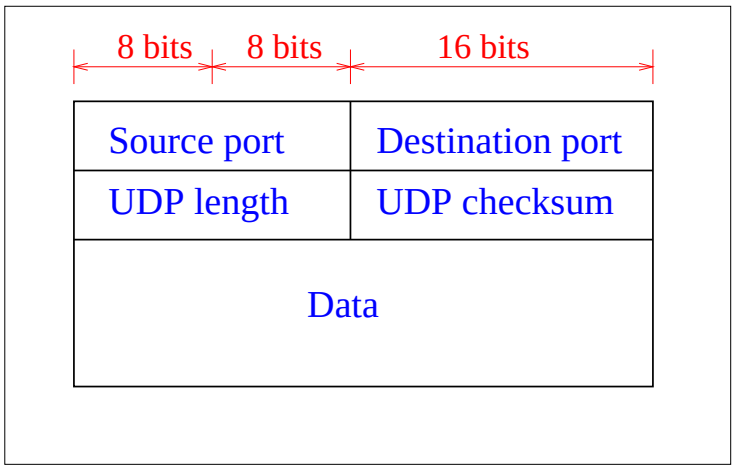
Congestion:

There is some effort in TCP to reduce congestion. TCP handles both congestion caused by a low bandwidth network, and that caused by a *slow* receiver. The TCP transmission software maintains a congestion window size variable. When transmission starts, the TCP software increases the transmission segment size until either it is the same as the requested segment size, or it gets no response[9]. Whenever a timeout occurs, the congestion window size variable is halved.

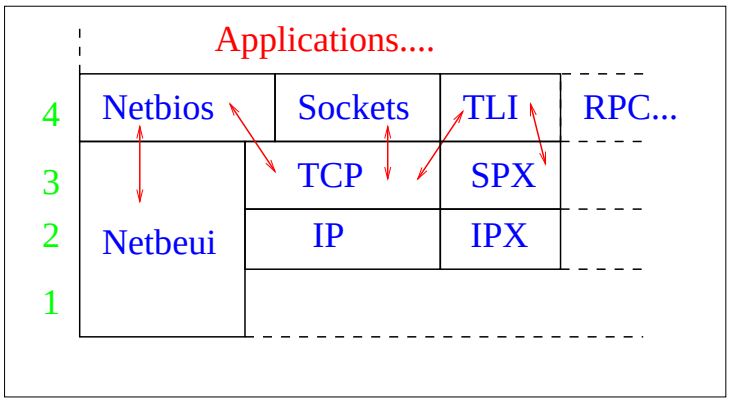
[9]Indicating that the link is congested, or perhaps doesn't support the packet size requested.

6.4.2 UDP

The UDP header is much simpler:



6.5 Session layer



All of the session layer protocols give similar service: they provide peer to peer service between processes on (possibly differing) machines.

6.5.1 Sockets

This is an old[10] API, available on all platforms. *WINSOCK* is a standard API to sockets for the PC/windows world. On UNIX and VMS sockets are accessed using a library (*libsocket.a*) and C

[10] In the computing world, anything over 10 years old is considered *old*!

calls. A socket address is composed of the IP address of the target machine and a port number identifying which process.

UNIX systems provide virtually all their services using sockets, using either TCP or UDP as a transport.

Question: How do two or more clients telnet to a system at the same time?

Answer: The first client connects to port 23, and then negotiates a high numbered port to do the session (10000) - it then gives up the port. The second client then connects to port 23 and negotiates a different port 10001.

The processes to handle services such as these are started as needed using the UNIX fork command. The code looks like this:

```
repeat
   x := IncomingConnection;
   result := fork ();
   if result = childprocess then
      processchild(x)
   else
      close(x)
until TheWorldComesToAnEnd!
```

The general method for using sockets is as follows:

- Before referencing the socket - create it:
  int socket(int sock_family, int sock_type, int protocol);
- Bind socket to a local address:
  int bind(int sock_descr, struct sockaddr *addr, int addrlen);
- Clients may issue a connect call:
  *int connect(int sock_descr, struct sockaddr *peer_addr, int addrlen);*
- Servers must listen:
  int listen(int sock_descr, int queue_length);
  and then accept incoming connections:
  *int accept(int sock_descr, struct sockaddr *peer_addr, int *addrlen);*

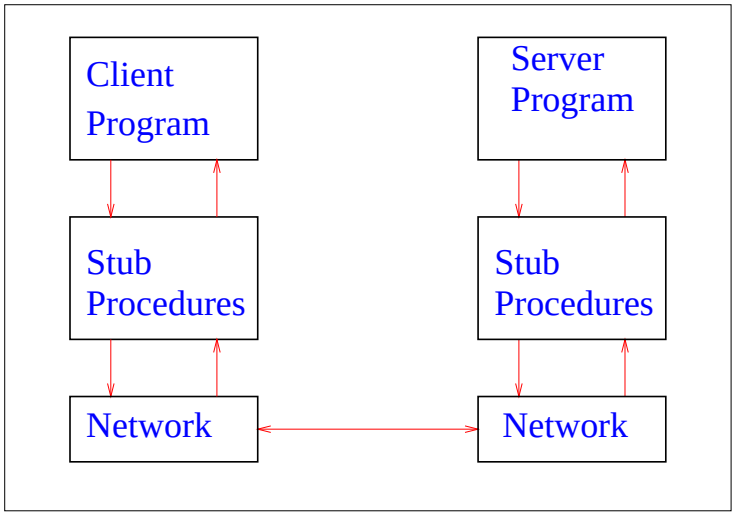
In Appendix C is an example of a small server using sockets.

6.5.2 RPC

The other session layer APIs give one level of service, but require careful programming. In particular, the *states* of each end of a link are defined only by the programmer.

> *The RPC model is that all interaction is like a procedure call.*

The RPC software is automatically generated by the *rpcgen* tool, which assures that call parameters are sent correctly and interprets the results.



- Client calls stub
- Stub formats arguments
- Stub uses network to *call*
- Stub receives *call*
- Marshalls/converts arguments
- *Calls* server

RPC calls

Main features:

- Parameter passing is call-by-value
- Must know location of server
- RPC has an exception mechanism to inform you of errors
- Idempotency: - If you call a procedure twice and it has the same effect as calling it once, it is called idempotent.

There are various call semantics available with RPC:

- Exactly once
- At most once
- At least once

Question: How does the *server* identify a procedure call over the network?

Answer: By number - A program id and a procedure number within that.

On *server* machines, there is often support software for registering RPCs. It is common (and easy) to use at least once semantics and idempotent (same strength) calls.

The usual way of writing RPC code is to:

- Design a series of sensible calls, with parameters and return data types.
- Specify those calls in an interface definition language.
- Use an IDL compiler (rpcgen) to generate client and server stub procedures.
- Link in those stubs with your client and server code.

Example:

```
Program DATE_PROG{
   version DATE_VERS {
      long   BM_DATE(void)   = 1
      string STR_DATE (void) = 2
   } = 1
} = 0
```

RPC is used by NFS (Network File System), AFS and WEBNFS. NFS uses idempotent calls and at least once semantics and is stable even with severe network problems. Sun and others are promoting WEBNFS as a fileserver for the internet.

6.6 Configuration & implementation

6.6.1 UNIX

UNIX systems normally have networking built into the kernel (SunOS), or loaded as needed using loadable kernel modules (IRIX, Linux). They all come with IP as the standard networking protocol. Other protocols are considered an extra, but often not difficult to add. IRIX and Linux systems both come with IP, SMB, Appletalk and Netware support *off the shelf*.

UNIX IP networking is built around the *inetd* daemon. When inetd is started at boot time, it reads its configuration information from /etc/inetd.conf and listens for connections on specified internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request.

After the program is finished, it continues to listen on the socket.

The relevant configuration files[11] are:

- **/etc/rpc** - maps RPC program numbers.
- **/etc/protocols** - maps IP protocol numbers to their number in the IP header.
- **/etc/services** - maps internet services to TCP and UDP port numbers
- **/etc/inetd.conf** - maps internet services to software to handle them.

6.6.2 DOS and Windows redirector

DOS was developed without computer networks in mind, so network additions are grafted into the operating system. The core of this added facility is the *network redirector.*

When I/O calls are made the redirector examines the call and if it is intended for the local machine it passes it to the I/O subsystem. If it is intended for the network it passes it to the network software.

To install networking software on a DOS machine we must:

1. Install NIC
2. Install NIC driver software
3. Install network layer software
4. Install Application layer (file server client)

[11] In typical UNIX style, configuration files are stored in the /etc directory.

5. Install the redirector

To uninstall we remove the network software elements in the reverse order.

Detail: The mechanism used to perform OS calls in DOS is the Software Interrupt (not the subroutine call). We use these calls instead of subroutine calls for two reasons:

- the INT call preserves registers
- very fast way of doing an indirect call (through a table).

The call:

- Make INT call
- All registers pushed on stack
- A vector address is fetched.
- Execution continues.

The return:

- Execute a RT1 (return from INT)
- Registers Pulled from Stack
- Program continues

When we add the redirector, it overwrites the INT vector table with its own address and also keeps track of the old IO addresses.

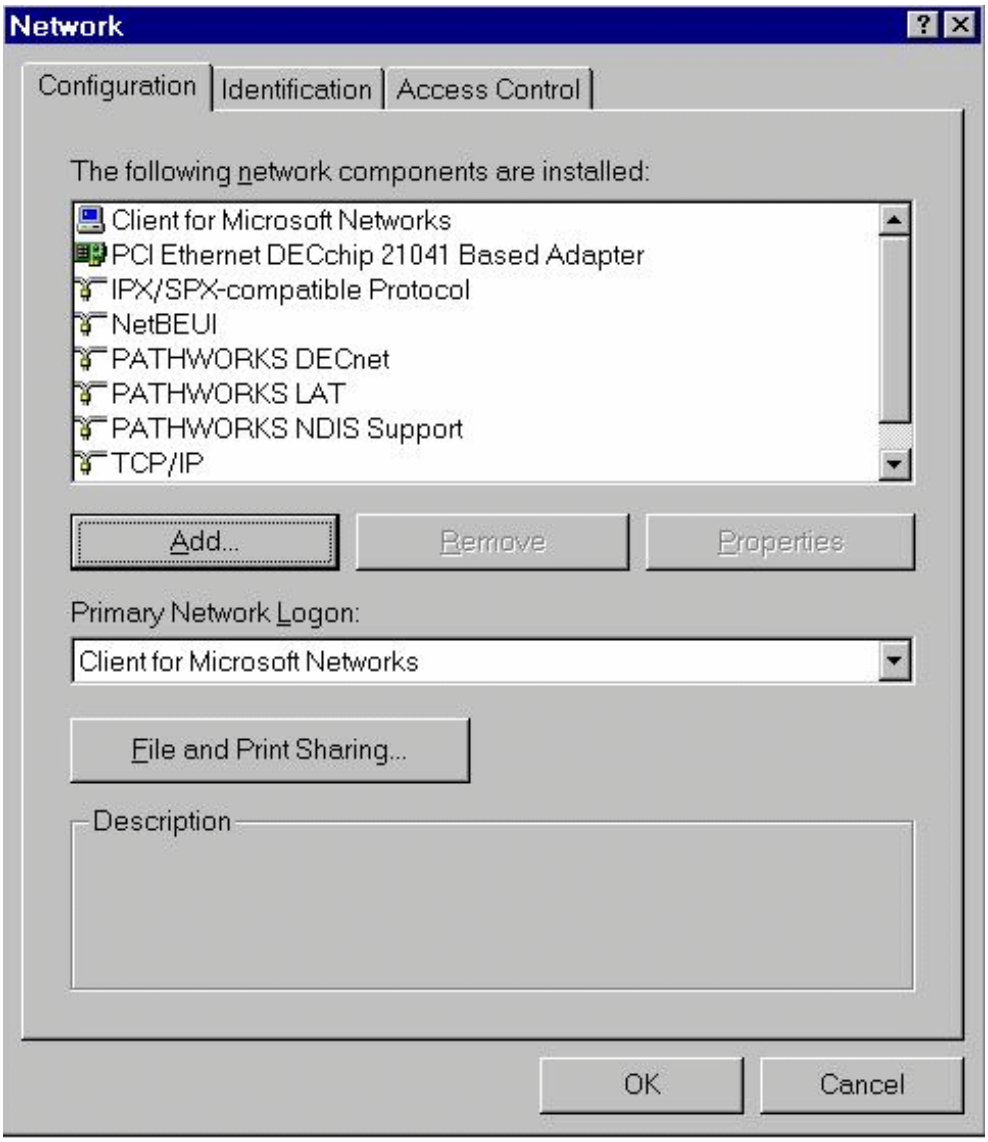
Question: Can we easily add other network protocols?

Answer: No. - PC network cards are not constructed to be driven by multiple unrelated protocols.The way in which we run multiple protocols is by providing a software *shim* between (single tasking) ethernet card/card drivers and multiple protocol stacks.

6.6.3 Win95/NT

In Win95, networking is still an *addon*, but better integrated than in WfW, or Win3.1. In NT the networking is *built in* as in UNIX, and much more reliable. NT configuration is done through the *Network* control panel, which allows you to select:

- Application layer networking software
- Protocols
- Interfaces



6.7 Diagnostic tools

At these layers, there are so many protocols, standards, and abstractions that it is a bit hard to identify diagnostic tools.

- **Inference** - we can infer from the behaviour of our application software some properties of the transport or session layers. If the lower layers were tested and work, but the software doesn't, then perhaps these layers are faulty.
- **Low level monitoring** - various software tools such as *sniff* and *tcpdump* can be used to monitor and check the sequence of transfers on a port by port basis.
- **Satan** - is a tool which will check a remote system for ports in use.

Chapter 7

Higher layers

The presentation layer is concerned with the syntax of the data transferred.

The application layer provides the user interface to the network wide services provided. Normally this layer provides the operating system interface used by user applications.

7.1 Sample standards

SNMP - The *Simple Network Management Protocol* is a protocol for remote network management. It is commonly used to monitor the behaviour of remote systems.

SMB - *Server Message Blocks* are a Microsoft/IBM protocol for file servers, commonly known as *Windows networking*.

NCP - The *Netware Core Protocols* are Netware's file system protocols.

DNS - The *Domain Name Service* supports the naming of entities on an IP network.

DES,RSA - The *Data Encryption Standard,* and *Rivest Shamir and Adelman's* encryption method are commonly used to ensure security on computer networks.

Standard	Application area	Protocols
SNMP	Network management	UDP/IP
SMB	File Server, printing	Many
NCP	File Server, printing	SPX/IPX, TCP/IP
DNS	Name resolving	UDP/IP
DES, RSA	Encryption	Any

7.2 Addressing

7.2.1 NCP

In the Netware world, servers have names, which may be up to 14 characters long. Objects within servers have names as well:

- **STAFF** - the STAFF server
- **STAFF/Hugh** - the Object ID Hugh on server STAFF.

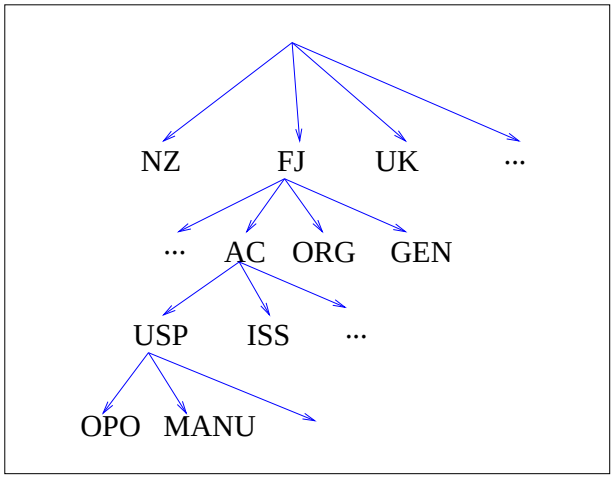
This naming scheme does not expand well. (How many STAFF servers are there in the world?)

7.2.2 IP and the DNS

Any IP active interface can have a name attached to it. In addition groupings (or regions) of interfaces can have names attached to them:

1. A host may have one or more *names*.
2. A *name* might not refer to a machine.
3. Hosts don't have to have *names*.
4. You cannot tell by *looking* at an IP name if it refers to a *host* or a *domain*.

The IP name space is hierarchical:



usp.ac.fj - is this the machine **usp** inside **ac.fj** or is it the domain **usp.ac.fj**?

Note Machines may be named in several different ways. For example:

- kai.ee.cit.ac.nz
- ee.cit.ac.nz
- www.ee.cit.ac.nz

All the above refer to the same machine. This naming scheme is extendable, and much more useful than the netware scheme.

Question: How is the namespace administered?

Answer: With a linked hierarchy of servers especially for name information:

1. A machine asks for namespace information about another machine. (Such as: is it a real name? What is its network address? Can I email to it?)
2. The machine asks the local DNS server.
3. If the local server doesnt know, it asks its parent server and so on.
4. The response updates all the intermediate machines, and the server caches the information for some time.

7.2.3 MacIntosh/Win 95/NT

All support a two-level naming scheme, involving a machine and a domain or zone.

Mac machines can dynamically acquire names and domains or they can be preset. The domain names map to the networks and are preset by the router.

NT machines can either use IP naming systems or Microsoft's own naming scheme based on single server technology.

7.3 Encryption

Security and Cryptographic systems act to reduce failure of systems due to the following threats:

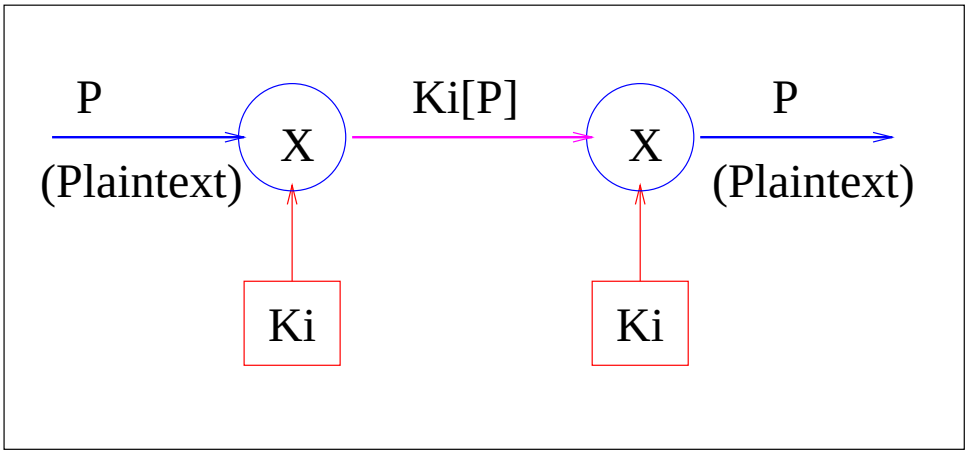
Interruption - attacking the availability of a service (Denial of Service).

Interception - attacks confidentiality.

Modification - attacks integrity.

Fabrication - attacks authenticity. Note that you may not need to decode a signal to fabricate it - you might just record and replay it.

7.3.1 Shared Keys:



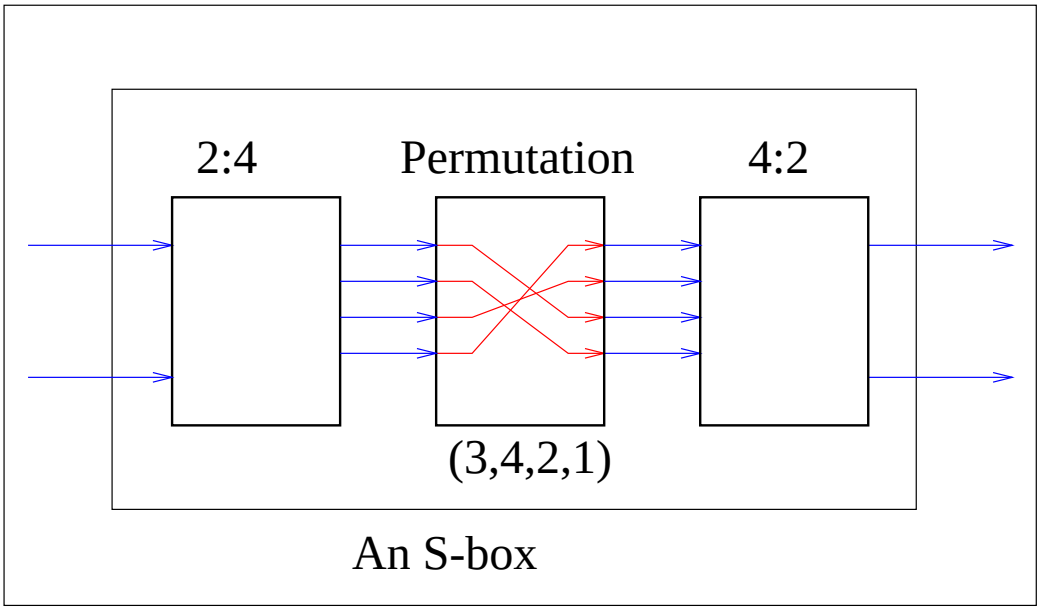
Shared key systems are generally considered inadequate, due to the difficulty in distributing keys.

7.3.2 Ciphertext

These systems encode the input stream using a substitution rule:

Code	Encoding
A	Q
B	V
C	X
D	W
...	...

The S-box (Substitution-Box) encodes *n* bit numbers to other *n* bit numbers and can be represented by the permutation. This is an S-box:



Ciphertext is easily breakable, particularly if you know the likely frequency of each of the codes. In the English language, the most common letters are:

- E T A O N I S H R D L U

(From most to least common).

7.3.3 Product Ciphers

We have seen two types of cipher: If you use both types at once, you have a product cipher which is generally harder to decode, especially if the P box has differing numbers of input and output lines (1 to many, 1 to 1 or many to 1).

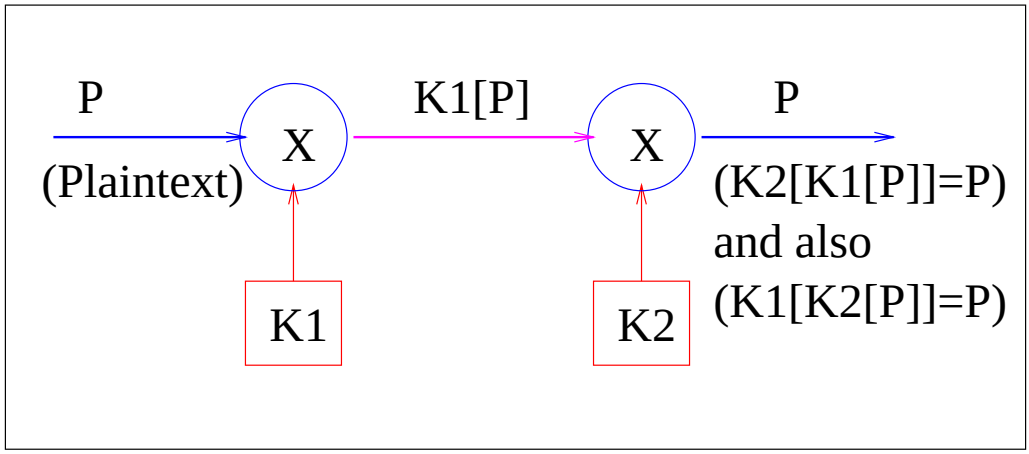
7.3.4 DES - Data Encryption Standard

DES was first proposed by IBM using 128 bit keys, but its security was reduced by NSA (the National Security Agency) to a 56 bit key (presumably so they could decode it in a reasonable length of time). At 1ms/GUESS. It would take 10^{80} years to solve 128 bit key encryption. The DES Standard gives a business level of safety, and is a product cipher.

The (shared) 56 bit key is used to generate 16 subkeys, which each control a sequenced P-box or S-box stage. DES works on 64 bit messages.

Note: If you intercept the key, you can decode the message. However, there are about 10^{17} keys.

7.3.5 Public key systems



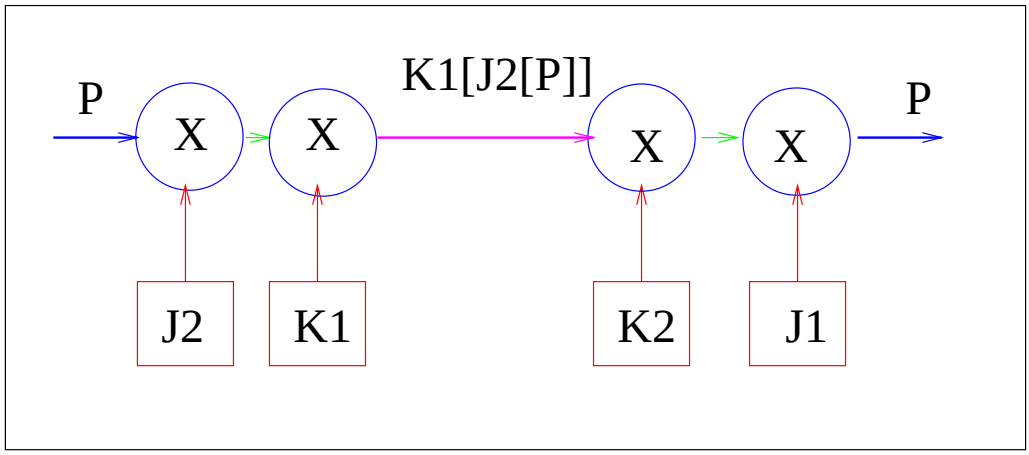
A machine can publish K1 as a public key, as long as it is not possible to create K2 from K1.

Authentication

We can use this to provide authentication as well.

If one machine wants to authentically transmit information, it encodes using both its private key and the recipient's public key:

The second machine uses the others public key and its own private key to decode.



RSA (Rivest, Shamir, Adelman)

This public key system relies on the properties of extremely large prime number to generate keys.

To create public key K_p:

1. Select two large primes P and Q.
2. Assign $x = (P-1)(Q-1)$.
3. Choose E relative prime to x. (This must satisfy condition for K_s given later)
4. Assign $N = P * Q$.
5. K_p is N concatenated with E.

To create private (secret) key K_s:

1. Choose D: $mod(D * E, x) = 1$.
2. K_s is N concatenated with D.

We encode plain text P by:

1. Pretend P is a number.
2. Calculate $c = mod(P^E, N)$.

To decode C back to P:

1. Calculate $P = mod(C^D, N)$.

We can calculate this with:

```
c := 1;  { attempting to calculate mod(P^Q,n) }
x := 0;
while x<>Q do
   begin
      x := x+1;
      c := mod(C*P,N)
   end;
{ Now C contains mod (P^Q,N) }
```

7.4 SNMP & ASN.1

In RPC, we met XDR, the external data representation for transferring data and agreeing on its meaning. This process is normally considered to lie in the presentation layer (layer 6). Not many standards directly relate to layer 6, but there is ISO8824-ASN[1]. ASN.1 defines an agreed syntax between layer 6 entities.

We define the abstract syntax using a language (ASN.1). We define the transfer syntax using a set of Basic Encoding Rules (BER).

Here is an example:

```
employeename ::= string
employeeage ::= integer
person ::= sequence {
   num Integer
   name string
   married Boolean
}
```

or (tagged)

```
person ::= sequence {
   num [Application] integer
   name [1] string
   married [2] Boolean
}
```

With these tagged items, which may contain other tagged items, we can specify an item by giving the list of tags (as either numbers or identifiers).

There is a worldwide naming system for ASN entities. It starts:

- iso.org.dod.internet....

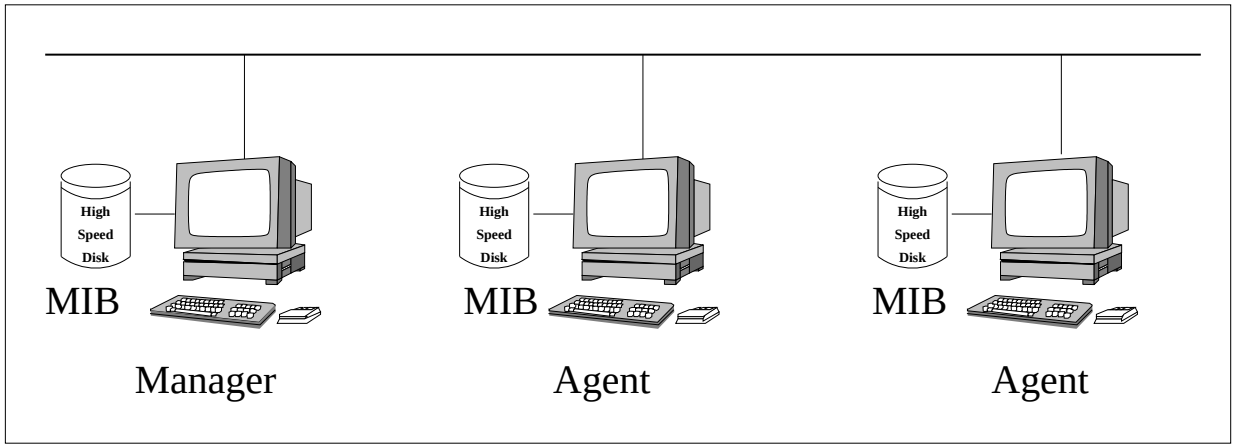
When using ASN.1 for management of remote systems we use

- iso.org.dod.internet.mgmt....

For example:

[1] ASN is **A**bstract **S**yntax **N**otation.

- iso.org.dod.internet.mgmt.mib.system.sysDescr.156.59.209.1



The most well known application for ASN.1 is in remote management, using SNMP. The *manager* is a *client*, the managed hosts are *servers*. Each SNMP entity has a MIB[2], which describes the items being *'served'*. MIBs define:

- How data will be transferred using the implied encoding rules.
- What data may be retrieved.

Manufacturers of ASN.1/SNMP capable equipment supply the MIBs describing their equipment for free.

7.5 Diagnostic tools

nslookup:

```
opo 51% nslookup manu
  Name:    manu.usp.ac.fj
  Address: 144.120.8.10
opo 52% nslookup
> set querytype=MX
> kai.ee.cit.ac.nz
  kai.ee.cit.ac.nz preference = 20, mail exchanger = ara-
iahi.cit.ac.nz
  kai.ee.cit.ac.nz preference = 10, mail ex-
changer = kai.ee.cit.ac.nz
opo 53%
```

[2]A MIB is the **M**anagement **I**nformation **B**ase.

smbtools:

```
opo 58% smbclient -L opo
  Server time is Sun Oct 11 23:56:22 1998
  Timezone is UTC+12.0
  Domain=[LANGROUP] OS=[Unix] Server=[Samba 1.9.17p2]
  Server=[OPO] User=[hugh] Workgroup=[LANGROUP] Domain=[LANGROUP]
      Sharename    Type      Comment
      ---------    ----      -------
      archive      Disk      Archive
      AST-PS       Printer
  This machine has a browse list:
      Server       Comment
      ---------    -------
      OPO          Samba 1.9.17p2
      PC0757       Ganesh Chand, Eco, SSED Rm
  This machine has a workgroup list:
      Workgroup    Master
      ---------    -------
      LANGROUP     OPO
opo 60% nmblookup manu
  Added inter-
face ip=144.120.8.248 bcast=144.120.11.255 nmask=255.255.252.0
  Sending queries to 144.120.11.255
  144.120.8.10 manu
opo 61%
```

Chapter 8

Application areas

8.1 File serving

We partition file serving systems into:

- Peer to peer, and
- Server centric.

Netware is normally considered to be server-centric. Netware servers are not used as *workstations*, and Netware clients don't *serve*.

All Macintosh AFS, WfW (SMB) and NFS systems can normally be either servers or clients. This is called *peer to peer* networking.

8.1.1 Netware

Netware have the largest share of the PC file server market. The product comes in 3 flavours:

v2.xx Now considered obsolete, but still in use in 30% of netware sites. V2.xx was written in assembler, and has not had updates for many years.

v3.xx Most widely used (50%) and most widely used NOS for PCs. The server must run on X386 processors (or better) because of addressing requirements, and the processor mode in which it runs.

v4.xx The latest version.

Netware 3.xx

1. 80386 (up to 4 GB ram possible)
2. Disk up to 32TB (1TB = $10^{12} = 2^{40}$ bytes)
3. Files up to 4GB (1GB = $10^9 = 2^{30}$ bytes)
4. Up to 100,000 files open at one time.
5. 2,000,000 directory entries/volume.
6. 16 NICs
7. Multi-protocol - allowing Netware servers to serve files to UNIX (NFS) , PCs and Macs (Appletalk/AFS), and as well route between connected networks.
8. Server backup locally, or over the network
9. Remote management. Netware servers can be totally managed remotely - even over a modem.
10. NLMs (Netware Loadable Modules). New functions can be loaded and unloaded at any time.

Note that the file systems expected on the different platforms are quite different:

- DOS systems expect 8.3 length names, and lack *permissions*.
- UNIX systems expect long file names, with a range of permissions
- NTFS systems expect long file names, with a different range of permissions
- Macs expect 64 character long file names, with resource and data forks.

File extensions found in a Netware system:

- **NLM** - System management and server functions. TCPIP.NLM is TCP/IP software, MONITOR.NLM is system monitoring software.
- **NCF** - Like DOS BAT files - used for Netware configuration.
- **DSK** - Disk driver software
- **LAN** - NIC driver software.
- **NAM** - Name space modules.

When you run a Novell file server, DOS is thrown away, and NOS becomes the OS. While running NOS on a file server, you cannot run DOS utilities. The only way to run DOS is to exit NOS.

1. Boot DOS.
2. Run a DOS program called server.exe, which loads and runs NOS.
3. Once it is running you can only use NOS commands.

Netware will only serve using its own file system. A DOS formatted disk cannot be served by Netware. You can either:

- Partition your disk(s) - 10 MB for DOS (Bootable, with server.exe) - Rest for NOS, or
- Use a separate floppy to boot the server. (The advantage of this is that you take the floppy away, marginally increasing the server's physical security.)

Netware allows forward and backward slashes for UNIX compatibility. The naming system is DOS like:

- **DOS:** A:\A\B\C.TXT
- **NOS:** STAFF/A:\A\B\C.TXT

Memory usage in Netware:

Netware systems cache disk accesses (read and write) for speed. In general, the more memory the better. You may calculate memory as follows:

- 2MB to get started.
- For serving to DOS: $M = 0.023 * \frac{DiskSize\,(MB)}{BlockSize\,(kB)}$ MB.
- For serving to MAC/UNIX: $M = 0.032 * \frac{DiskSize\,(MB)}{BlockSize\,(kB)}$ MB.
- Round up to next power of 2
- Adding NLMs requires more memory, up to 2MB per NLM. Once you have added a NLM, the *monitor* program can indicate what memory it is using.

Example: 200MB disk in 4k blocks and 4GB in 8k blocks:

- $M_{200} = 0.023 * \frac{200}{4} = 1.15$ MB, and
- $M_{4000} = 0.032 * \frac{4000}{8} = 11.5$ MB.
- So total memory needed is 16MB.

The more extra memory you have, the more available for caching with its consequent speed up.

Note: Caching means that copies are kept in memory of either what **is** on the disk or what **should** be on the disk. If the power fails, some material that is cached may be lost.

- If this is of concern, either use an UPS[1], or turn off caching.

Novell File System Basic Structure:

All Novell boxes must have a volume called SYS with the following four directories:

- **Sys:\login** - Anything for public access (boot config, login.exe, etc)
- **Sys:\public** - All the netware utilities available for *all* logged-in users
- **Sys:\system** - Supervisory and maintenance programs
- **Sys:\mail** - Mail and individual start up files

In addition to these 4 directories you will need user data directories and application directories.

Novell access Rights:

Novell's file access security is oriented toward the *users* not the *file*s. Users are granted sets of permissions to read, write, delete and search files and directories. Novell uses the term *Trustee* for a user and the rights assigned are called *Trustee Assignments* . Novell also has groups which may be assigned rights and then users may be made to belong to groups (inheriting all the group rights).

[1] **U**ninterruptable **P**ower **S**upply

Novell commands which support rights:

- **Right** - View system rights.
- **Grant** - Enable access to specified files, directories.
- **Revoke** - Remove trustee assignments.
- **Filer** - Menu driven file/user group management facility
- **Syscon** - General administration.

Booting a Novell File Server:

First boot DOS, then execute SERVER.EXE. It expects the following configuration files:

STARTUP.NCF - On DOS partition. A little like the DOS CONFIG.SYS. If it does not exist SYS will not be mounted. The main use is to get SYS mounted.

AUTOEXEC.NCF - Should be in Sys:\system. A little like AUTOEXEC.BAT. It loads NICs, NLMs and so on.

The Novell NOS prompt is : (colon). The most common command is LOAD:

- **In STARTUP.NCF:**
  : LOAD ISADISK
  : MOUNT SYS
- **In AUTOEXEC.NCF:**
  : LOAD NE2000 NAME=ETHERNET INT=5 DMA=3 ....
  : BIND IPX TO ETHERNET
  : MOUNT APPS
  : MOUNT USERS

First Time:

The first time you run Novell, you are asked for an internal number. You can choose any number except one you have used on another Netware server on your network.

8.1.2 SMB and NT domains

NT is a multi-tasking operating system similar to VMS. It comes in two flavours:

NT Workstation - Limited to 10 inbound connections. Single RAS (modem) service. Up to 2 processors.

NT Server - Unlimited inbound connections. 256 RAS (modem) services. Up to 4 processors.

NT systems are normally set up to work in either a *workgroup* or a *domain* environment.

Workgroup

This is a scheme in which a collection of related computers (eg sales, marketing, admin) share resources such as disks, printers and modems. Each computer in a workgroup manages its own accounts and access policies. Often users manage their own workgroup.

- Easy to set up.
- Not manageable at large sites.

Domain

The machines share a common data base of accounts and security (access policies). A single NT server is set up to act as the final say on accounts and policies. This machine is called the PDC[2]. There can be only one PDC for a domain. NT's domains do not correspond to networks or IP subnets. A single domain can span many networks, and multiple domains can coexist on a single subnet.

- Managed by system administrator.
- Suitable for larger networks.

[2] **P**rimary **D**omain **C**ontroller

Administration

In the *administrative tools* folder is a utility for adding users and groups.You can specify such items as:

- User name, login name, comment
- Groups the user belongs to
- Home directory
- Password policy
- Allowable login times
- User environment

By Default NT has the following groups

- **GUEST** - Limited access to resources.
- **USERS** - Default rights for an end user (will be able to run applications and save files).
- **POWER USER** - User access and other limited system management functions.
- **ADMINISTRATOR** - Complete control.
- **REPLICATOR** - Backup - specific rights for system administration.

NT File Systems

DOS/FAT - Maximum File Size 4GB
Partition Size 4GB
Attributes Read only/Archive/System/Hidden
File name length 255 characters

- **Note:** FAT files can be undeleted easily, have a minimal disk overhead (1M/0.5%), and are most efficient with a disk size under 200 MB. FAT file systems cannot be protected by NT.

HPFS - Maximum File Size 4GB
Partition Size 2TB (Disk geometry limits to 8 GB)
Attributes RO, A,S,H, Extended (User specified)

- **Note**: Long HPFS file names are not visible to DOS applications. HPFS has poor performance over 400 MB, cannot be protected by NT, and has an overhead of about 2MB. Files cannot be undeleted.

NTFS - Maximum File Size 16 EB (An Exabyte[3]is 2^{60} Bytes).
Partition Size 16 EB
NTFS has extended attributes that can grow, including file creation, time modified and so on.

- **Note:** NTFS is a *journalled* file system, and files cannot be undeleted. It has a high overhead (5 MB/5%), with a minimum disk size of 50 MB (i.e. no floppies). NTFS is good at keeping fragmentation low.

8.1.3 NFS

The **N**etwork **F**ile **S**ystem, allows a client workstation to perform transparent file access over the network. Using it, a client workstation can operate on files that reside on a variety of servers, server architectures and across a variety of operating systems. Client file access calls are converted to NFS protocol requests, and are sent to the server system over the network. The server receives the request, performs the actual file system operation, and sends a response back to the client.

The Network File System operates in a stateless fashion using remote procedure (RPC) calls built on top of external data representation (XDR) protocol.

The RPC protocol provides for version and authentication parameters to be exchanged for security over the network. A server can grant access to a specific filesystem to certain clients by adding an entry for that filesystem to the server's */etc/exports* file and running *exportfs*.

A client gains access to that filesystem with the *mount* system call.

Internals

NFS normally 'sits on top of' Unix file systems, which typically have extremely large file sizes and spaces. They are mostly journalled, with an overhead proportional to the disk size (1 to 10%). UFS file systems are good at defragmentation, and it is common for the file systems to have behavioural extensions.

- XFS is a *speed guaranteed* file system found on IRIX machines.

NFS based file server systems use *secure writes*. That is, if a workstation writes something to the server, it waits for an acknowledgement that the write is complete before continuing. This is nearly always slower than insecure writes, due to propagation delays. In a test of identical PC-based systems we determined the following speeds:

File Server	Read speed	Write speed	Mode
Netware	500 KB/s	500 KB/s	Insecure
UNIX	500 KB/s	300 KB/s	Secure
NT	300 KB/s	180 KB/s	Secure

[3]The prefixes are Kilo, Mega, Giga, Tera, Peta, Exa, Zetta and Yotta for 2^{10}, 2^{20}, 2^{30}, 2^{40}, 2^{50}, 2^{60}, 2^{70} and 2^{80} Bytes.

8.2 Printing - LPR and LPD

The LPR/LPD system is a protocol specifically for printing. Clients use LPR to submit print jobs to a queue. The LPD process is the print server. LPR and LPD clients and servers are available for all systems and provide a simple way to make network printers available.

Network printers always have LPR/LPD as one of the standard printing protocols.

8.3 Web services

The spread of Internet web browsers to the desktop cannot be ignored. Modern browsers are large complex applications with the capability of playing a significant role in distributed applications.

Originally the web consisted of hypertext only (linked text and pictures) - however the demand for interactive content led first to:

- *Forms* - the CGI (Common Gateway Interface) standard provides semi-interactive web pages.

and then to:

- *Java* - (and Javascript) interpreted byte code, which runs at the client.

Java distributed objects (called *applets* or little applications) are just one step towards a CORBA-like distributed environment. (We still however need a support ORB infrastructure). Various Java-builder suppliers provide CORBA ORBs for your Java applications.

8.3.1 Java

Java is:

- Object oriented
- Interpreted - bytecode, but JIT[4]compilation gives C/C++ speed.
- Multi-platform
- A lot more than fancy web pages.

Java supports the development of robust, secure and distributed applications, and can be used in one of two main ways:

[4]JIT stands for **J**ust **I**n **T**ime compilation. The byte code is compiled on the fly as it is downloaded into an executable form on the local machine.

- If your code has a main(), you can run it directly:

```
opo> javac Mycode.java
opo> java Mycode
```

In Appendix B is an example of a small client/server application.

With no *main()*, you may embed an applet in a web page. Use HTML like:

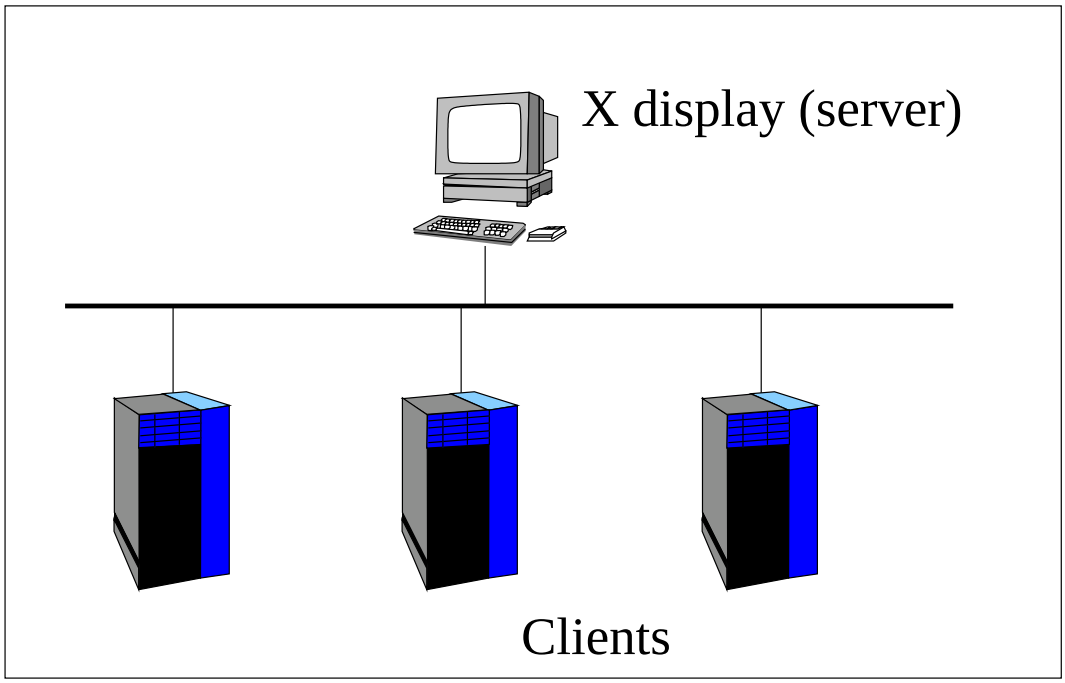
```
<applet code="Mycode.class" width=150 height=150> </applet>
```

Here are some local URLs with help info:

- **http://opo/java/release.html** - The Java SDK API and commands.
- **http://opo/Jdev/qtour.html** - A quick tour around the Cosmo Java development environment.
- **http://opo/javascript/index.html** - Stuff on Javascript.
- **http://opo/java/nutshell** (Sample code from the nutshell book)
- **http://opo/java/tutorial** (Sun's Java tutorial)
- **http://opo/~hugh/archive/lang/java** (My small archive of stuff!)

8.4 X

In the 1980s, various groups around the world were developing a distributed window system. One of the more successful developments at Stanford University was the **W** window system, and when MIT began to develop their window system, they used *what came after* **W** (X)! The X window system allows graphics to be distributed efficiently over a network.



In the **X** view of the world, the *display* is the center, displaying graphical material from one or more clients, which can be distributed about the network. The X architecture involves:

- **X servers** - which display the graphical information.
- **X display managers** - controlling logging in and out.
- **X window managers** - controlling the *look and feel* of the display.
- **The X protocol** - an application layer protocol.
- **X client programs** - distributed about the network.

X provides a *mechanism* to do things, it seldom sets *policy*. There are few limits to its behaviour - you can have whatever window decorations you like, and a range of display types. Your client programs will work regardless.

X is commonly used to display UNIX desktops, but there are products that allow your X displays to access NT, Win95 and Macintosh machines[5]. These systems are less useful than you might think, because none of these systems are multiuser.

X is:

[5]The machines run software that catches all display events, and convert them to X protocol messages.

- Mature,
- Well defined,
- Efficient,
- Stable,
- Available on all platforms,
- Useable with text, graphics, video and sound.

In the UNIX/IP world, high numbered ports (6000) and UDP is used for transport.

8.5 Thin clients

The X window system needs a fairly large computer at the display[6]. Thin client technology allows you to run *small* code on the display, communicating using a *small* protocol with a server on a larger machine. The server can worry about efficiently managing the display.

Another use for these systems is to allow dial-up *takeover* control of a remote machine.

8.5.1 WinFrame & WinCenter

Are extensions to NT that:

- Make NT multiuser,
- Display on remote machines using X, and
- Display on remote machines using proprietary thin client protocols.

Example: A large PC (say a dual PII) with a large amount of memory (say 384MB) can manage 30 or more 286 computers each with only 2MB of memory. Each of these computers acts like a very fast Pentium, unless everyone simultaneously starts compiling!

8.5.2 VNC

VNC[7] is a thin client system developed at the Olivetti and Oracle Research Lab, and recently made available to the Internet community under the GNU General Public License. It provides:

- Servers for UNIX, Win95, NT, Macintoshes... and
- Clients for UNIX, Win95, NT, Macintoshes and a range of other peculiar machines!

VNC does not give you multiuser NT. However it appears stable.

[6] A Pentium machine, or a Sparc should be sufficient. At least 8MB memory would be required. The more memory the better!

[7] **V**irtual **N**etwork **C**omputing.

8.6 Electronic mail

SMTP[8] is a mail transfer scheme found on the Internet. Its architecture consists of similar mail *servers,* which are set up to store and forward electronic mail. You can test mail forwarding and routing by forcing paths into the e-mail addresses:

- hugh%opo.usp.ac.fj@waikato.ac.nz

SMTP uses a single simple TCP connection on port 25 to transfer mail. You can test mail server operation by just connecting to the server port:

- telnet opo 25, or..
- telnet opo smtp

SMTP mail systems add fields to the mail messages:

```
Return Path : ...
Received    : ...
Organisation: ...
Date        : ...
etc, etc.
```

SMTP mailers are commonly eight bit clean, but there is no guarantee, so when you send binary files, they should be encoded. Unfortunately there is not a single encoding standard, and most mail readers have to have various decoders to make sense of incoming data.

[8] **S**imple **M**ail **T**ransfer **P**rotocol.

Chapter 9

Other topics

9.1 ATM

ATM[1] technology is a cell based transport system. Transmitted data is broken into fixed 53 byte cells, which can be switched and routed efficiently. The ATM header has only 5 bytes, leaving 48 bytes for data.

ATM networks revolve around an ATM switch, which can route full speed traffic between nodes.

9.2 CORBA

The main features that we expect from an *object* are:

- **Encapsulation** - data and functions to operate on them.
- **Inheritance** - main construction method for new software modules (rather than aggregation).
- **Polymorphism** - object's behaviour can change at run time.

A distributed object system involves

- objects (components) which are distributed, and
- brokers - which name, identify and support the objects.

[1] In this context, ATM stands for **A**synchronous **T**ransfer **M**ode, not automatic teller machines.

It is common now to find distributed objects performing as:

1. TP monitors
2. CSCW support tools
3. servers of various sorts
4. databases
5. applications

In addition, distributed objects have made it to the desktop (most notably in the form of Java applets - processor independant, reasonably efficient and safe). If we expect our objects to interoperate successfully in a distributed system, we have to agree on a set of standards. The two (incompatible) standards are:

1. OMG's CORBA
2. Microsoft's DCOM/OLE

CORBA object interfaces are specified using yet another IDL (Interface Definition Language). The IDL is language and implementation independant, and specifies object interfaces unambiguously. CORBA has been adopted by every operating system supplier except for Microsoft - who are promoting a different standard. However CORBA object extensions for Microsoft products are available from third parties.

CORBA has two principal components in addition to your application objects:

1. The ORB
2. Standard objects and facilities

ORB:

An ORB can be viewed as the 'bus' interconnecting objects. It:

- defines the the handling of all communication between objects.
- manages the identification/naming of each object.

A mandatory requirement for CORBA implementations is the IIOP - a TCP/IP ORB protocol. This ensures at least a minimum level of interoperability. You can of course use other protocols - such as DCE's RPC - which may give security and encryption features.

Standard Objects:

CORBA has a set of pre-defined standard objects for use in building new applications:

- Naming
- Transactions
- Time
- SQL
- Security
- Licensing
- Documents (OpenDoc <=> OLE)

9.3 DCOM/OLE

- **DCOM** - Distributed Component Object Model.
- **OLE** - Object Linking and Embedding.

OLE is an object-oriented environment for components, which has gone through several transformations, but is supplied as a local service with Win95.

Microsoft's consistent stated goal is to provide all OS and applications as OLE components.

Note: ActiveX is a minimal OLE object found on the Internet.

9.4 NOS

The generic term 'Network Operating System' (NOS) describes a uniform view of the services on a network. A NOS may involve all sorts of middleware running on different platforms, but tying them together with agreed protocols.

Example: - We may use a global time service - accessable from all systems on the local network. It is common to use NTP or a similar product to ensure clock synchronicity within an enterprise.

The NOS functions are often provided on-top-of or bundled with conventional OS's such as NT or UNIX.

Note: The NOS view of the enterprise computing environment may become obsolete. Web services, and distributed object technologies are doing similar things, and have already spread to the desktop.

9.4.1 DCE:

The Distributed Computing Environment from OSF has all the elements of an enterprise NOS. It uses the folowing key technologies:

- RPC
- Naming
- Time
- Security
- DFS
- Threads

And where do we find DCE?

- In Transarcs TP monitor
- In H.P.'s CORBA - ORB Plus
- In uSoft's Network OLE
- In turnkey UNIX systems

DCE RPC

The DCE RPC is borrowed from Hewlett Packard, and has the following properties:

DCE RPC's have an IDL and compiler. The DCE RPC has more developed authentication than the (Sun) RPC. The DCE RPC can dynamically allocate servers. It is protocol independant.

In addition, the DCE RPC is integrated with the DCE security and naming services.

Naming

DCE has adopted X.500 for naming services. Any system resource can be identified by name using a DNS-like distributed hierarchical name service.

For example:

- files servers
- print queues

And also the not-so-conventional:

- programs
- processes

Like the DNS, names can be grouped, defined and administered locally.

Time

All systems eventually synchronise to external time providers, through local time servers. The local time servers keep track of each others time as well and correct errors.

There must be at least three local time servers.

Security

DCE uses Kerberos[2]. Kerberos is generally considered the most secure authentication system. It was designed by deeply suspicious people.

DCE uses Kerberos for

- authentication
- maintaining a user database
- authorization.

The user database is kept on a secure (normally dedicated) server.

All communication is done using encrypted authenticated RPC - systems wishing to communicate validate keys using the security server - which they trust. All keys invalidate themselves after a short time.

Note: DCE's security system is a model for other NOS's.

DFS file system

The DCE file system (DFS) is based on AFS - the Andrew File System from CMU (Carnegie Melon University in the U.S.). DFS files are location independant, and are often cached when a workstation accesses the file.

Modifications to a cached copy are invisibly propagated to other copies of the file. In addition, you can mirror files across a network (again invisibly) to either:

- Ensure fast response in a geographically dispersed environment
- Ensure high availablity (if one of the servers fails).

[2]Kerberos is the three headed dog that guards the gates of hell!

List of Figures

List of Tables

Appendix A

ASCII table

Dec	Hex		Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	^@	*nul*	32	20	*spc*	64	40	@	96	60	‘
1	01	^A	*soh*	33	21	!	65	41	A	97	61	a
2	02	^B	*stx*	34	22	“	66	42	B	98	62	b
3	03	^C	*etx*	35	23	#	67	43	C	99	63	c
4	04	^D	*eot*	36	24	$	68	44	D	100	64	d
5	05	^E	*enq*	37	25	%	69	45	E	101	65	e
6	06	^F	*ack*	38	26	&	70	46	F	102	66	f
7	07	^G	*bel*	39	27	’	71	47	G	103	67	g
8	08	^H	*bs*	40	28	(	72	48	H	104	68	h
9	09	^I	*ht*	41	29	)	73	49	I	105	69	i
10	0A	^J	*lf*	42	2A	*	74	4A	J	106	6A	j
11	0B	^K	*vt*	43	2B	+	75	4B	K	107	6B	k
12	0C	^L	*ff*	44	2C	,	76	4C	L	108	6C	l
13	0D	^M	*cr*	45	2D	-	77	4D	M	109	6D	m
14	0E	^N	*so*	46	2E	.	78	4E	N	110	6E	n
15	0F	^O	*si*	47	2F	/	79	4F	O	111	6F	o
16	10	^P	*dle*	48	30	0	80	50	P	112	70	p
17	11	^Q	*dc1*	49	31	1	81	51	Q	113	71	q
18	12	^R	*dc2*	50	32	2	82	52	R	114	72	r
19	13	^S	*dc3*	51	33	3	83	53	S	115	73	s
20	14	^T	*dc4*	52	34	4	84	54	T	116	74	t
21	15	^U	*nak*	53	35	5	85	55	U	117	75	u
22	16	^V	*syn*	54	36	6	86	56	V	118	76	v
23	17	^W	*etb*	55	37	7	87	57	W	119	77	w
24	18	^X	*can*	56	38	8	88	58	X	120	78	x
25	19	^Y	*ew*	57	39	9	89	59	Y	121	79	y
26	1A	^Z	*sub*	58	3A	:	90	5A	Z	122	7A	z
27	1B	^[	*esc*	59	3B	;	91	5B	[	123	7B	{
28	1C	^\	*fs*	60	3C	<	92	5C	\	124	7C	|
29	1D	^]	*gs*	61	3D	=	93	5D	]	125	7D	}
30	1E	^^	*rs*	62	3E	>	94	5E	^	126	7E	~
31	1F	^_	*us*	63	3F	?	95	5F	_	127	7F	*del*

Appendix B

Java code

```
This example is taken from ’Java in a Nut-
shell’ by David Flanagan:
The Client:
import java.io.*; import java.net.*;
public class Client { public static final int DE-
FAULT_PORT = 6789; public static void usage() { Sys-
tem.out.println("Usage: java Client <host-
name> [<port>]"); System.exit(0); }
public static void main(String[] args) { int port = DE-
FAULT_PORT; Socket s = null; if ((args.length != 1)&&(args.length !=
age(); if (args.length == 1) port = DE-
FAULT_PORT; else { try port = Inte-
ger.parseInt(args[1]); catch (NumberFormatExcep-
tion e) usage(); } try { //Cre-
ate socket s = new Socket( args[0],port); //Cre-
ate streams DataInputStream sin = new DataInput-
Stream(s.getInputStream()); PrintStream sout = new PrintStream(s.getO
sole... DataInputStream in = new DataInput-
Stream(System.in); Sys-
tem.out.println("Connected to: " + s.getInetAddress() + ":" + s.getPo
tem.out.print("> "); Sys-
tem.out.flush(); line = in.readLine(); if (line == null) break; //Sen
tem.out.println("Connection closed by server"); break; } Sys-
tem.out.println(line); } } catch (IOException e) Sys-
tem.err.println(e); fi-
nally { try if (s != null) s.close(); catch (IOExcep-
tion e2); } } }
The Server:
```

```
import java.io.*; import java.net.*;
public class Server extends Thread { public fi-
nal static int DEFAULT_PORT = 6789; pro-
tected int port; protected ServerSocket lis-
ten_socket; // Exit with an error message when an ex-
ception occurs pub-
lic static void fail(Exception e, String msg) { Sys-
tem.err.println(msg + ":" + e); System.exit(1); }
//Create a server socket to listen on -
start thread pub-
lic Server(int port) { if (port==0) port = DE-
FAULT_PORT; this.port = port; try { lis-
ten_socket = new ServerSocket(port); } catch (IOExcep-
tion e) fail(e, "Exception creat-
ing server socket"); System.out.println("Server: lis-
tening on port " + port); this.start(); }
//Loop forever, accepting connections - creating con-
nection //object to handle new socket pub-
lic void run() { try { while (true) { Socket client_socket = lis-
ten_socket.accept(); Connection c = new Connec-
tion(client_socket); } } catch (IOExcep-
tion e) fail(e, "Exception while listening"); }
//Start the server pub-
lic static void main(String[] args) { int port=0; if (args.length ==
teger.parseInt(args[0]); catch (NumberFormatExcep-
tion e) port = 0; } new Server(port); } }
//Thread to handle comms with client class Connec-
tion extends Thread { protected Socket client; pro-
tected DataInputStream in; protected PrintStream out;
//Initialize streams, start thread public Connec-
tion(Socket client_socket) { client = client_socket; try { in = new D
put-
Stream(client.getInputStream()); out = new PrintStream(client.getOutp
ception e) { try client.close(); catch (IOExcep-
tion e2); System.err.println("Exception while get-
ting socket streams: " + e); return; } this.start(); }
//The service: reverse a line pub-
lic void run() { String line;
StringBuffer revline; int len; try { for (;;) { //read in line line =
verse it len = line.length(); revline = new String-
Buffer(len); for (int i=len-1;i>=0;i--
) revline.insert(len-1-
i,line.charAt(i)); //Write it out out.println(revline); } } catch (IO
```

```
ception e); finally try client.close(); catch (IOExcep-
tion e2); } } This exam-
ple is taken from ’Java in a Nutshell’ by David Flana-
gan:
The Client:
import java.io.*; import java.net.*;
public class Client { public static final int DE-
FAULT_PORT = 6789; public static void usage() { Sys-
tem.out.println("Usage: java Client <host-
name> [<port>]"); System.exit(0); }
public static void main(String[] args) { int port = DE-
FAULT_PORT; Socket s = null; if ((args.length != 1)&&(args.length !=
age(); if (args.length == 1) port = DE-
FAULT_PORT; else { try port = Inte-
ger.parseInt(args[1]); catch (NumberFormatExcep-
tion e) usage(); } try { //Cre-
ate socket s = new Socket( args[0],port); //Cre-
ate streams DataInputStream sin = new DataInput-
Stream(s.getInputStream()); PrintStream sout = new PrintStream(s.getO
sole... DataInputStream in = new DataInput-
Stream(System.in); Sys-
tem.out.println("Connected to: " + s.getInetAddress() + ":" + s.getPo
tem.out.print("> "); Sys-
tem.out.flush(); line = in.readLine(); if (line == null) break; //Sen
tem.out.println("Connection closed by server"); break; } Sys-
tem.out.println(line); } } catch (IOException e) Sys-
tem.err.println(e); fi-
nally { try if (s != null) s.close(); catch (IOExcep-
tion e2); } } }
The Server:
import java.io.*; import java.net.*;
public class Server extends Thread { public fi-
nal static int DEFAULT_PORT = 6789; pro-
tected int port; protected ServerSocket lis-
ten_socket; // Exit with an error message when an ex-
ception occurs pub-
lic static void fail(Exception e, String msg) { Sys-
tem.err.println(msg + ":" + e); System.exit(1); }
//Create a server socket to listen on -
start thread pub-
lic Server(int port) { if (port==0) port = DE-
FAULT_PORT; this.port = port; try { lis-
ten_socket = new ServerSocket(port); } catch (IOExcep-
```

```
tion e) fail(e, "Exception creat-
ing server socket"); System.out.println("Server: lis-
tening on port " + port); this.start(); }
//Loop forever, accepting connections - creating con-
nection //object to handle new socket pub-
lic void run() { try { while (true) { Socket client_socket = lis-
ten_socket.accept(); Connection c = new Connec-
tion(client_socket); } } catch (IOExcep-
tion e) fail(e, "Exception while listening"); }
//Start the server pub-
lic static void main(String[] args) { int port=0; if (args.length ==
teger.parseInt(args[0]); catch (NumberFormatExcep-
tion e) port = 0; } new Server(port); } }
//Thread to handle comms with client class Connec-
tion extends Thread { protected Socket client; pro-
tected DataInputStream in; protected PrintStream out;
//Initialize streams, start thread public Connec-
tion(Socket client_socket) { client = client_socket; try { in = new D
put-
Stream(client.getInputStream()); out = new PrintStream(client.getOutp
ception e) { try client.close(); catch (IOExcep-
tion e2); System.err.println("Exception while get-
ting socket streams: " + e); return; } this.start(); }
//The service: reverse a line pub-
lic void run() { String line;
StringBuffer revline; int len; try { for (;;) { //read in line line =
verse it len = line.length(); revline = new String-
Buffer(len); for (int i=len-1;i>=0;i--
) revline.insert(len-1-
i,line.charAt(i)); //Write it out out.println(revline); } } catch (IO
ception e); finally try client.close(); catch (IOExcep-
tion e2); } }
```

Appendix C

Sockets code

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define SERV_TCP_PORT 9000
#define MAXLINE 512

process(sockfd)
int sockfd;
{
  int n;
  char c;
  for ( ; ; ) {
    n = read(sockfd,&c,1);
    if (n==0) return;
    else
      if (n<0)
        printf("process: readline error\n");
    switch (c){
    case 'Q': return;break;
    case 'N': write(sockfd,"N result\n",9);break;
    case 'P': write(sockfd,"P result\n",9);break;
    }
  }
}


main(argc, argv)
int argc;
char *argv[];
{
   int     sockfd,newsockfd,clilen;
   struct sockaddr_in      cli_addr,serv_addr;

   if ( ( sockfd=socket(AF_INET, SOCK_STREAM,0))<0)
      printf("server: can't open stream socket\n");
   bzero((char *) &serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
   serv_addr.sin_port =htons(SERV_TCP_PORT);
   if (bind(sockfd,(struct sockaddr *) &serv_addr, sizeof(serv_addr))<0)
      printf("server: can't bind local address\n");
   listen(sockfd,5);
   for ( ; ; ) {
      clilen = sizeof(cli_addr);
      newsockfd = accept(sockfd,(struct sockaddr *) &cli_addr,&clilen);
      printf("New connection....\n");
      if (newsockfd<0)
         printf("server: accept error\n");
      write(newsockfd,"Hugh's Server\n",14);
      process(newsockfd);
      close(newsockfd);
   }
}
```