G52CCN Revision Guide

Craig Knott & Joe Nash

Version 1.4

May 15, 2013

With special thanks to Casey Modell, who sat with us during the writing of this guide, reminding us that, no matter how poorly we performed, our degree would still be worth more than one in Geography.

Further thanks to Luke Hovell and Adam Gask, for reading through the guide and pointing out all of the mistakes.

Contents

1 Data Transmission **3**
- 1.1 Transmission Media . . . 3
- 1.2 Local Asynchronous Communication and RS-232 . . . 3
- 1.3 The Three B Brothers; Bit rate, Baud rate and Bandwidth . . . 4
- 1.4 Long Distance Communications . . . 5
- 1.5 MODEM . . . 7
- 1.6 Multiplexing . . . 7

2 Packet Transmission **8**
- 2.1 Packets . . . 8
- 2.2 Hardware Frames . . . 8
- 2.3 Error Detection . . . 9
- 2.4 Local area networks (LANs) . . . 11
- 2.5 Carrier Sense Multiple Access . . . 13
- 2.6 Extending LANs . . . 13
- 2.7 Hardware addressing and Frame Type Identification . . . 15
- 2.8 Wide area networks (WANs) . . . 16
- 2.9 Protocols and Layering . . . 18

3 Internetworking **21**
- 3.1 Classless Addressing . . . 22
- 3.2 Addressing methods . . . 24
- 3.3 IP Datagrams . . . 24
- 3.4 The Future of IP . . . 25
- 3.5 Transport Control Protocol (TCP) . . . 26
- 3.6 Connectionless Or Connection Oriented? . . . 27

4 Network Applications **28**
- 4.1 Client - Server Paradigm . . . 28
- 4.2 Distributed Programming . . . 29
- 4.3 The Socket Interface . . . 30
- 4.4 Network Security . . . 32
- 4.5 IP Security & IPSec . . . 32
- 4.6 DNS Security Issues . . . 33
- 4.7 Firewalls . . . 33
- 4.8 Intrusion Detection Systems . . . 33
- 4.9 Vulnerability Assessment and Honeypots . . . 34
- 4.10 Example Network Applications . . . 34

5 In Closing **36**

A Dijkstra's Algorithm **38**

B Distance Vector Routing **38**

1 Data Transmission

1.1 Transmission Media

Twisted Pair Cable
This cable has two wires twisting around each other. Each wire has a plastic coating that insulates the two wires and prevent electric current from flowing between them. The twisting makes the wires less susceptible to electromagnetic radiation. However, the twisting wires have problems with especially strong electrical noise, close physical proximity to the source of noise, and high frequency used for communication.

Co-Axial Cable
The co-axial cable aims to counter the problems that the twisted pair cable has. The cable has a wire, surrounded by plastic insulation, braided metal, and a secondary plastic covering. This forms a flexible cylinder around the wire, that also provides a barrier to electromagnetic radiation from any direction.

Fiber Optic Cable
These are thin strands of glass or transparent plastic. Typically fiber optics are used for single direction communications; one end connects to a laser or LED, and the other end connects to a photosensitive device. When two way communication is required, two fibers are used, one for each direction.

1.2 Local Asynchronous Communication and RS-232

There are three broad categories of transmission: *synchronous*, which is where transmissions occur continuously, with no gaps; *isochronous*, which is where transmissions occur at regular intervals with a fixed gap between each; and *asynchronous*, which is where transmissions can occur at any time, with an arbitrary delay between each [8].

We will focus solely on asynchronous communication; in which there is no coordination between the sender and receiver prior to transmission - the receiver does not know when the sender will transmit (for instance, a keyboard connected to a computer, or a user clicking a link on a website)[16]. This means that data is sent whenever it is ready; leading to varying delays between transmissions. The transmitter must include additional information to each transmission, to signify the start and end of the data being sent.

RS-232[1] is one standard for specifying asynchronous communication. RS-232 specifies that: the physical connection length must be less than 50 feet, electrical voltage must range from -15 to +15 volts, and the line coding is such that negative voltage corresponds to a logical 1 value, and a positive voltage corresponds to logical 0. RS-232 specifies that each set of data being transmitted (generally a single character of 7 or 8 bits), must start with a 0 bit, and end with a 1 bit[2]. The idle state is represented as a 1 bit, so the reciever can tell when a transmission begins - using the 0 start bit. This gives an additional data overhead of 2 bits, per 7/8 bits sent, shown in figure 1.

[1] Formally known as RS-232-C
[2] Known as the phantom bit

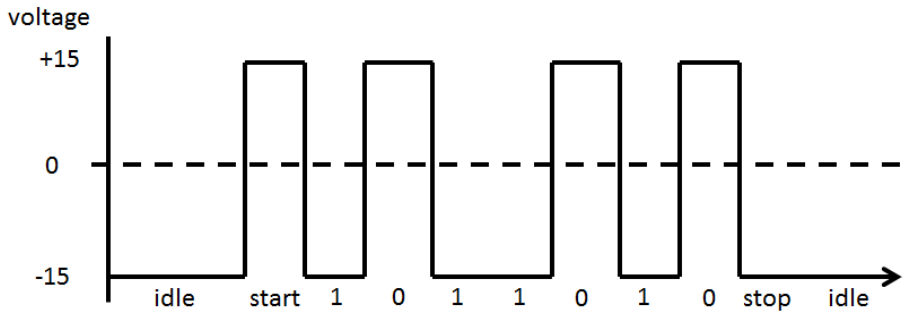


Figure 1: Illustration of voltage during transmission of a 7-bit character using RS-232. Specifically how the characters "1011010" would be transmitted

RS-232 uses a 25 pin connector, which transmits on pin 2 and receiver on pin 3 (the others are used for control functions). As with all hardware, there are physical limitations. For instance, hardware cannot instantly change voltage and, as such, there is a maximum speed at which bits can be sent, and imperfect signals can occur. RS-232 specifies how much tolerance there should be for these imperfect signals.

1.3 The Three B Brothers; Bit rate, Baud rate and Bandwidth

The rate at which data can be sent through a network, measured in bits per second, is known as *bit rate*. The transmission hardware on these networks is rated in *baud rate*, which is the number of signals they can generate per second. In RS-232, these two rates are specified to be equal. The baud rate of RS-232 is configurable, but the sender and receiver must agree of a length of time that each bit is held, and the maximum number of bits per second (differences in baud rate can cause framing errors). Finally, *bandwidth* is the number of signals per second that a specific medium can accommodate.
There are two main theorems that relate to these three ratings the Nyquist Sampling Theorem (figure 2) and Shannon's Theorem (figure 3).

$$D = 2\ B\ \log_2 K$$

Figure 2: Nyquist's Sampling Theorem.

Where D is Data Rate (bps), B is bandwidth (in Hz), and K is level of signals. This theorem specifies the theoretical limit on the maximum speed that data can be sent over an error-free medium. This theorem encourages engineers to explore ways to encodes bits on a signal, that would allow for more bits to be transmitted per unit time.

$$D = B\ \log_2(1 + \frac{S}{N})$$

Figure 3: Shannon's Theorem.

Where D is Data Rate (bps), B is bandwidth, and $\frac{S}{N}$ is the signal to noise ratio (db). Shannon's theorem deals with real world situations, where mediums are likely to have noise. It informs engineers that no amount of clever encoding can overcome the laws of physics - there is a fundamental limit on the number of bits per second that can be transmitted. It's worth noting that the signal to noise ratio is usually give in decibels, but this must be converted to bits per second, to be used within the formula. The formula for this is $db = 10 \log_{10} \frac{S}{N}$. An example of this conversion is displayed overleaf, for 30db.

$$
\begin{aligned}
& 30 = 10 \log_{10} \frac{S}{N} \\
= & \quad \{ \quad \text{applying division} \quad \} \\
& 3 = \log_{10} \frac{S}{N} \\
= & \quad \{ \quad \text{applying rules of logarithms} \quad \} \\
& 10^3 = \frac{S}{N} \\
= & \quad \{ \quad \text{applying power and rearrange} \quad \} \\
& \frac{S}{N} = 1,000 \ bps
\end{aligned}
$$

For a full example of using these formulae, consider this question. How fast can data be sent across a voice telephone call where the capabilities of the telephone system are a bandwidth of 3,000Hz, and a $\frac{S}{N}$ of 30dB.

$$
\begin{aligned}
& D = B \ \log_2(1 + \frac{S}{N}) \\
= & \quad \{ \quad \text{substituing known values} \quad \} \\
& D = 3,000 \ \log_2(1 + \frac{S}{N}) \\
= & \quad \{ \quad \text{using decibel to bps conversion of signal noise ratio} \quad \} \\
& D = 3,000 \ \log_2(1 + 1,000) \\
= & \quad \{ \quad \text{arithmetic} \quad \} \\
& D = 3,000 \ \log_2(1,001) \\
= & \quad \{ \quad \text{applying logarithms} \quad \} \\
& D \approx 3,000 * 10 \\
= & \quad \{ \quad \text{arithmetic} \quad \} \\
& D \approx 30,000 \ bps
\end{aligned}
$$

When working with logarithms, remember $\log_a x = N \implies a^N = x$

1.4 Long Distance Communications

Signalling across long distances has many problems associated with it, for instance, the loss of signal due to resistance in wires [17]. Long distance communications use a *carrier*, which is a continuously oscillating electro-magnetic wave - this is because this signal will propagate further than other signals. The system modifies the carrier to represent different information, this is called *modulation*. There are three main forms of modulation; frequency, amplitude and phase shifting[23].

Amplitude modulation uses changes in amplitude to represent different bit values. Two different amplitudes are used to represent the 0 bit and the 1 bit, respectively - as shown in figure 4.

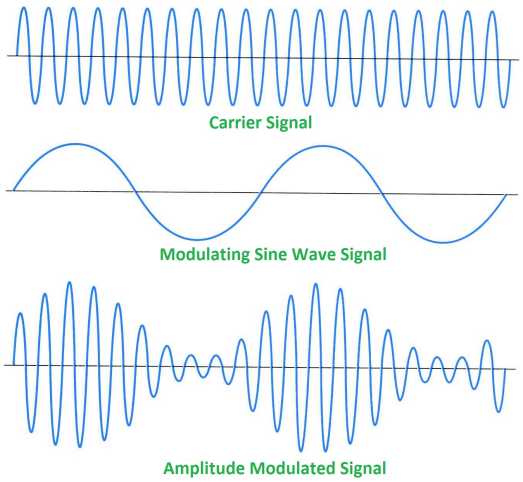


Figure 4: Amplification modulation on an oscillating wave (analogue signal) [3]

Frequency modulation uses changes in frequency to represent different bit values. Each frequency represents either 0 or 1, shown in figure 5.

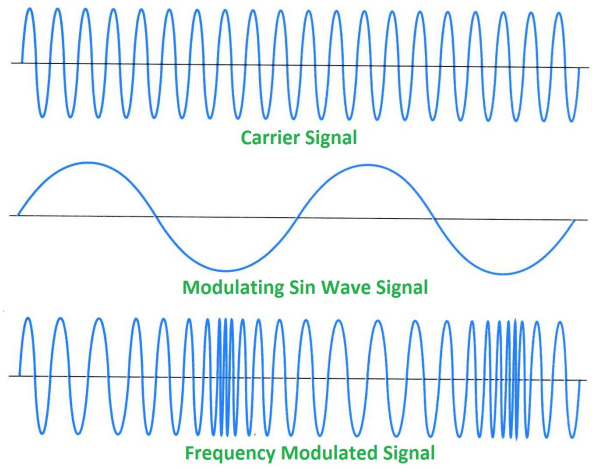


Figure 5: Frequency modulation on an oscillating wave (analogue signal) [1]

The final method, phase shift modulation, is slightly more complex. There are two main varieties of phase shift modulation, that we will refer to as *simple*, and *complex*. Simple phase shifting involves oscillating 360/720° for each bit. If the next bit is different from the previous bit, the next wave will be flipped. This concept is demonstrated in figure 6.

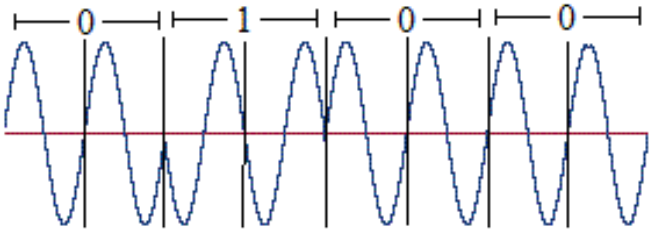


Figure 6: Simple Phase Shift modulation on an oscillating wave

The more complex version of phase shifting involves shifting a certain degree of the wave, depending on the *two* bits to be represented. This is dependant on the key that is being used; but each key must specify what bits are represented by each degree of shift (for instance, no shift represents 00, 1/4 shift represents 01, 1/2 shift represents 10, and 3/4 shift represents 11).

1.5 MODEM

A MODEM, which stands for Modulator/Demodulator, is a way to implement fully duplex communications. The modulator end takes bits and modulates them, and the demodulator end takes a carrier wave and produces bits. This duplex communication is displayed in figure 7.

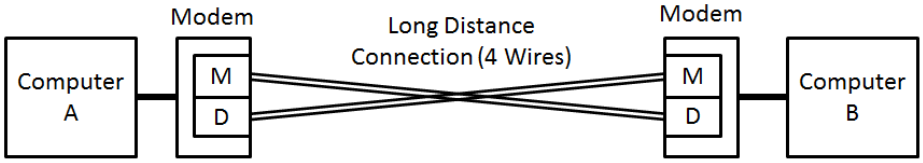


Figure 7: Fully duplex communication between two sites. Each requiring a modulator and demodulator

The three types of MODEMs are dial-up, radio and optical. Fun fact; the strange noises that dial-up MODEMs made was the process of converting from a digital signal to audio.

1.6 Multiplexing

Multiplexing is used for communicating between multiple sources of information, over one shared medium (see figure 8). The *multiplexor* is a mechanism that implement the combination of information, and the *demultiplexor* is the mechanism that separates the information back into it's individual streams.

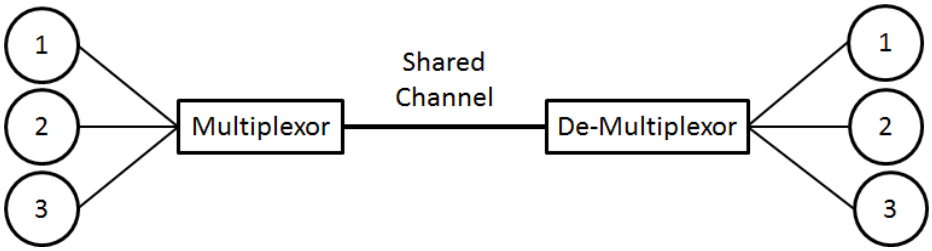


Figure 8: Multiplexing in action

There exist four basic approaches to this multiplexing:

- Frequency Division Multiplexing
  In this method, each signal is assigned a different channel; which is represented by a different carrier frequency. The demultiplexor applies a set of filters that each extract a small range of frequencies near one of the carrier frequencies. It's important to note that, because the FDM mechanism can separate the frequency from others without otherwise modifying the signal, any modulation technique can be applied. Unfortunately, there is a limit imposed on the number of frequencies that can be used for channels, because if two channels are too close, interference can occur.

- Time Division Multiplexing
  This method is an alternative to FDM in which the sources take turns in accessing the shared medium. This sharing is co-ordinated through round-robin scheduling[3]. This multiplexing is a broad concept, that appears in many forms.

[3]Fun fact; the name round robin comes from the French "Ruban", meaning "Ribbon". Specifically, when peasants used to write petitions to complain to the King, he would behead the first few names on the list. To combat this, the peasants signed their names in a circle, like a ribbon, such that no name on the list came first - i.e, each name had equal precedence.

- Wavelength Division Multiplexing
  This method is an extension of FDM, but instead used in optical figures. The signals are as wavelengths of light, denoted by λ, informally called *colors*. The shared medium is accessed through a prism, which is how the data is split up into these colours.

- Code Division Multiplexing
  The final form of multiplexing, used in parts of the cellular telephone system and for some satellite communications, is known as Code Division Multiplexing. This uses the mathematical idea of orthogonal vector spaces (don't worry, we don't need this for the exam!).

2 Packet Transmission

2.1 Packets

Packets are small blocks of data that are transmitted over networks. This helps to detect transmission errors, and gives fair access for a shared connection between many computer (i.e. one large file will not congest a shared medium)[19]. Networks that make use of packets are known as *packet networks*. These packet networks use a form of Time Division Multiplexing to organise sending information.

2.2 Hardware Frames

A frame is a specific format for wrapping a packet; which can be defined differently dependant on the hardware. An example of a very simple frame is shown in figure 9.

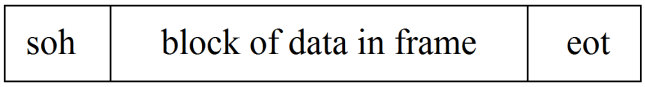


Figure 9: A simple example of the framing of a packet

The features of this frame format is that basic error detection is inbuilt. If the transmitter crashes, the "eot" bits will not arrive, and if the receiver crashes the "soh" bits mark the next valid frame. However, this means each block of data requires a 2 character overhead, and not all values can be carried in the data ("soh" and "eot" specifically).

To combat this second issue, we use *byte stuffing*. We use special characters in the data that signify the occurrence of "control characters" (like the "eot" and "soh" from the previous example). The transmitter scans the data block and replaces all occurrences of the control characters with their new escape characters; the receiver performs the opposite action. An example of byte stuffing can be seen in figure 10; in which "soh" and "eot" are still the beginning and end segements, respectively, but in this case, all "esc" are replaced with "esc+z", all in-text "soh" are replaced with "esc+x", and in-text "eot" are replaced with "esc+y".

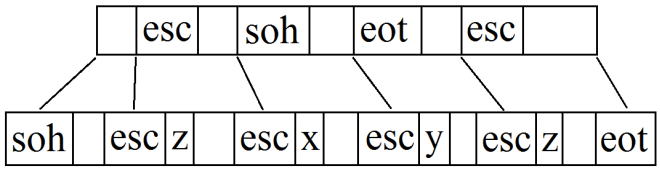


Figure 10: An example of byte stuffing.

2.3 Error Detection

Much of the complexity of networks arises from interference and transmission errors; which leads to data being lost or changed, or random data appearing. Errors come in two distinct fashions: single-bit, in which a number of single bits of data are modified; and burst errors, in which multiple sequential bits are changed (for instance, an electrical surge).

To correct errors, we must first define ways of detecting them. The general process of this is to include extra information that gives a summary of the original data, which the receiver can check against what they are expecting. From here we can decide to correct the errors, or retransmit the error ridden packets.

There are three main error detection methods that are important to know, and these are:

Parity bits and parity checking
In this method, an extra bit, the parity bit, is appended to the transmitted data. The value of this bit is determined by the number of 1 bits in the data, and the parity method being used. In *even* parity checking, a parity bit of 0 will be appended if the byte has an even number of 1 bits, and a 1 appended otherwise (for *odd* parity checking, these values are inverted - see figure 11).

Data	Number of 1 bits	Even Parity Checking	Odd Parity Checking
00000000	0	0	1
01011011	5	1	0
01010101	4	0	1

Figure 11: Data bytes and their corresponding parity bits

This method is simple, and only requires a small additional cost, but at the same time can only detect a limited number of error types.

Checksums
In this method, the data is interpreted as a sequence of integers, which are added together to create a single integer that represents the data - called a checksum. This checksum is appended to the frame, and is checked by the receiver. Generally these come in 16 or 32 bit forms, which lead to a data overhead of the respective size. This method produces very little computational overhead (only simple additions are required), but periodic reversal of bits cannot be detected (for instance, if the same index bit is flipped in every section, the checksum would be different, but the value of the data block would be changed). An example check sum can be seen in figure 12, and an example of undetected errors in checksumming can be seen in figure 13.

	H	e	l	l	o		w	o	r	l	d
+	48	65	6C	6C	6F	20	77	6F	72	6C	64
=											43C

Figure 12: An example checksum, on the string “Hello World”

Data Item	Check Sum	Data Item	Check Sum
0001	1	0011	3
0010	2	0000	0
0011	3	0001	1
0001	1	0011	3
Total	7	Total	7

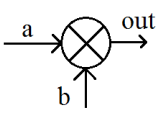
Figure 13: Two different data items with the same checksum (with every third bit being changed between the first set and the second)

Cyclic Redundancy Checks (CRC)
The final method of error detection is cyclic redundancy checking, which detects more errors than the other two methods, and only requires a simple extra piece of hardware to do so. Unfortunately for us, this is also the most difficult of the method to understand. There are three main components to CRC:

1. *D*, a piece of data that is *d* bits long, that is to be sent
2. *G*, a "generator", which is a predefined sequence of bits of length *r+1* (see below)
3. *R*, an additional sequence of bits, of length *r*, that is appended to D, such that the resulting sequence is exactly divisible by G, using modulo 2 arithmetic.

When the sender wishes to send *D*, the receiver and itself agree on the generator, *G*. The sender appends an extra sequence of bits, *R*, to *D*, resulting in a sequence that is exactly divisible by *G*, using modulo 2 arithmetic (aka, binary). The receiver divides the received bit pattern by *G* and checks whether the remainder is 0 or not. Any other values means an error in transmission has occurred. the advantages of this method is that burst errors (of less than *r+1* bits), and odd numbers of single-bit errors can be detected. Further to this, burst errors with length *greater* than *r+1* can be predicted with a probability of $1-0.5^r$.

There is an additional hardware component that is necessary to make this arithmetic possible - an XOR unit ($\otimes$), and a shift register. An XOR unit takes two inputs *a* and *b*, and returns 0 if the two values are equal, and 1 if they are different (shown in figure 14).



a	b	out
0	0	0
1	0	1
0	1	1
1	1	0

Figure 14: A simple XOR unit, with it's corresponding truth table

The shift register stores a small number of bits. When a new bit is "shifted" in, the one on the opposite is outputted (generally into an XOR unit). As mentioned, these two types of unit are used together in CRC. The system has bits fed in, one at a time, and they eventually will all pass through the XOR units. Once the process is complete, the system will be in the "final" state - the final output representing the CRC - which will be calculated by both sender and receiver. This process can be seen in figure 15.

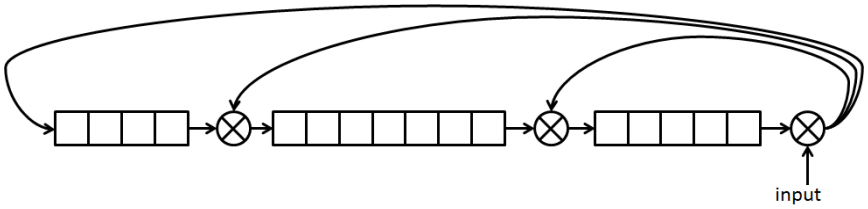


Figure 15: The entire CRC loop

CRC only has a data overhead of 16 or 32 bits, a combination of simple hardware devices produces a low computation overhead, and it is good for detecting burst errors. We need not worry about why CRCs work, but it should be intuitive that each single bit can dramatically affect the whole CRC due to its multiple entry points.

At this point we should evaluate a new packet frame, which improves upon the one shown earlier (figure 9), by adding byte-stuffing, and a CRC segment - shown in figure 16.

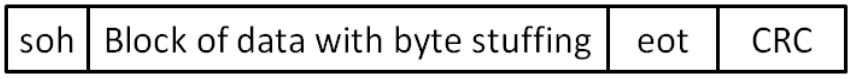


Figure 16: New frame with error detection and framing information

2.4 Local area networks (LANs)

There are many many different types of local area networks, and these different methods are called different "topologies". LANs connect more computer than any other form of network [15], due to "Locality of Reference" (the principle that computers are more likely to communicate with those nearby, and the same computers repeatedly).

Mesh Networks
An early network topology that is not used so much any more is the mesh network. This used dedicated links between each pair of computers (see figure 17). This was useful because hardware and frame details could be tailored for each link, and it was easy to enforce security and privacy. However, with every computer requiring connection to all others on the network, this topology is simply non-scalable (as $\frac{n^2-n}{2}$ connections are required).

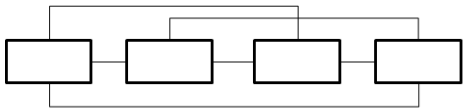


Figure 17: Example mesh network

Star Network
In this topology, all computers are connected to one central point - known as a hub (see figure 18). This is a somewhat robust topology, but the hub can become a bottleneck.

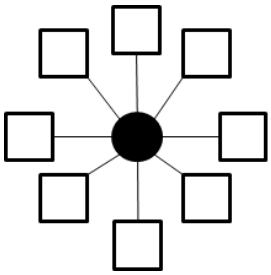


Figure 18: Example star network

Ring Networks
In a ring network (demonstrated in figure 19), the computers are connected in a closed loop. Each computer is connected to two others in this loop. A token is passed around this loop, that designates which computer is currently allowed to transmit. Data flows right around the ring, with the receiver making a copy when it is finally reached. The copy is passed around until the transmitter receives it, and can then check for any transmission errors (this is known commonly as the IBM Token Ring Network). This network enables easy coordination, but is sensitive to cables being cut. To combat this, more advanced ring networks have two wires, which, in case of emergency, can loop back onto itself to exclude a fault computer from the loop (figure 20) - this is a special example known as a FDDI.

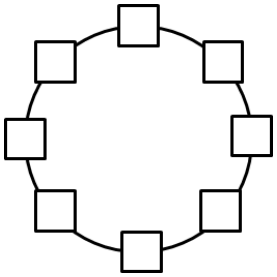


Figure 19: Example Ring Network

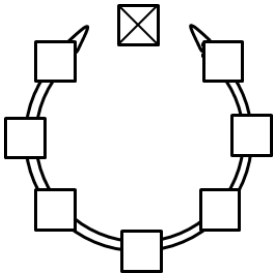


Figure 20: Backward looping due to a failed station in a ring network

BUS Networks
The final topology is BUS. In this topology, there is a single cable that each of the computers attach to (see figure 21). This means that less wiring is required for each computer, but the network is, once again, sensitive to a cable cut. Data is sent across the entire length of the cable, but only the destination computer acknowledges it; all others ignore it. A problem with this topology is that all of the computers on the cable can transmit and receive freely - which this raises the issue of collisions occurring. This issue is resolved with Carrier Sense Multiple Access - which is discussed in section 2.5

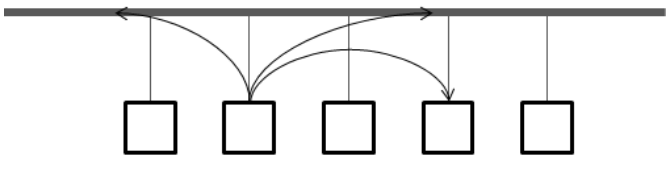


Figure 21: Example BUS network

2.5 Carrier Sense Multiple Access

Carrier Sense is the monitoring of the Ether [4], to detect signals in transmission. Multiple Access refers to computers only transmitting when they have determined the Ether is free - preventing other computers interrupting ongoing transmissions. This, however, does not prevent two computers transmitting at the same time - which can cause collisions. This means that computers must also sense for garbled transmissions, that are the result of a collision. This is known as *Collision Detection*, and is used in conjunction with CSMA to produce the CSMA-CD protocol. After a collision has occurred, the involved computers must wait before transmitting again. The time waited is random up to a specified max. This max is then doubled for each subsequent collision (known as binary exponential backoff). The reason that collision detection does not work on wireless mediums is (as quoted from Middlesex University London [4]):

"*This scheme does not work in wireless networks. The signal transmission in wireless networks is over the air rather along a wire. The strength of a signal decreases proportionally to the distance to the sender. The sender may apply carrier sense and detect an idle medium, thus the sender starts sending. But a collision happens at the receiver due to a second sender, which is out of the detection range of the first sender. This is the hidden terminal problem. Both the carrier sense (CS) and collision detection (CD) do not work. Thus, the standard MAC scheme CSMA/CD from wired network fails in a wireless scenario.*"

Another addition to CSMA is *Collision Avoidance*, which is used in situations where Collision Detection is ineffective. Collision Avoidance requires the sender to send a small request message to the receiver. If and only if the Ether is clear, the receiver will respond with a "Clear to Send" message, which blocks all other adjacent computers from sending. Collision avoidance is more appropriate for wireless networks.

A question to ask ourselves, is what should we do if the medium is busy? There are three answers to this:

- Non persistent CSMA (deferential)
  Wait a random time and try again.
- 1-persistent CSMA (selfish)
  Listen to the medium until it is idle, then transmit.
- p-persistent (compromise)
  Wait until idle, then either transmit or delay (each with equal probability).

2.6 Extending LANs

As the name suggests, *local* area networks have limitation on them regarding the distance they cover. Protocols such as CSMA/CD require time proportional to the length of the cable used [12]. Network designers must balance the capacity, delay, distance, and cost of a network when they are designing it.

There are some methods that can used to extend LANs, without any loss of information. This is generally done by using additional interface hardware that can relay signals across longer distances. These methods are discussed following:

[4] Shared network space

Fiber Optic Extensions
This method introduces an optical fiber, and a pair of fiber modems. These tools connect on one end to the Ethernet, and on the other, a computer. This simple concept is displayed in figure 22.

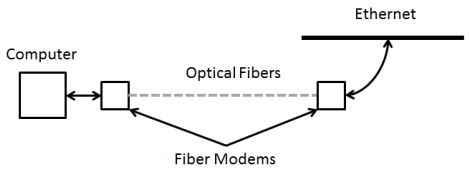


Figure 22: An example of a fiber optic extension

Repeaters
A *repeater* is an analogue device used to propagate LAN signals over long distances. A repeater does not understand packets or signal coding. Instead, it merely amplifies the signal receiver, and transmits this amplified version. This means that *all* signals are amplified, including collisions and noises, which severely limits their scalability. Repeaters were used much more with the original Ethernet, and nowadays are being introduced with infrared receivers to permit longer distance communication. Figure 23 shows an example extension. Note that, while multiple repeaters can be chained together, the Ethernet standard states that there should be no more than four repeaters between any two computers.

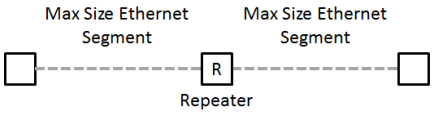
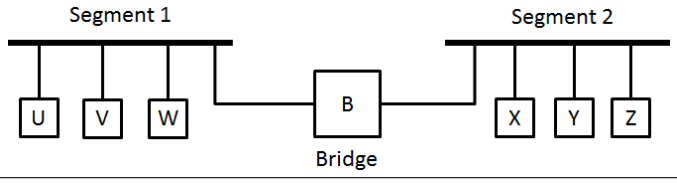


Figure 23: An example of a repeater being used to extend a connection

Bridges
A *bridge* is a mechanism that connects two segments of a LAN, and transfers packets between them. This is done through the use of *promiscuous mode*, which is where a network device can view every packet sent across the network. A bridge is more intelligent than a repeater, and will not forward erroneous frames, or frames that do not *need* forwarding (for instance, those with a destination address being on the same segment as their source address). As computers broadcast and transmit, the bridge learns which segments the individual computers are on (from a frame's source address). Figure 24 is an example of bridge learning[5].



Event	Segment 1 Knowledge	Segment 2 Knowledge
Bridge Boots	-	-
U sends to V	U	-
V sends to U	U, V	-
Z broadcasts	U, V	Z
Y sends to V	U, V	Z, Y
X sends to W	U, V	Z, Y, X
W sends to Z	U, V, W	Z, Y, X

Figure 24: An example of bridge learning

[5]This is a very simple concept, but will mostly likely be on the exam, so be sure to learn it!

Switches
An Ethernet switch (also known as a Layer 2 switch), is an electronic device that resembles a hub. Like a hub, a switch provides multiple ports that each attach to a single computer, and allows computers to send frames to one another. This produces a greater data rate, due to the parallelism that the switch provides. A conceptual model of a switch can be seen in figure 25.

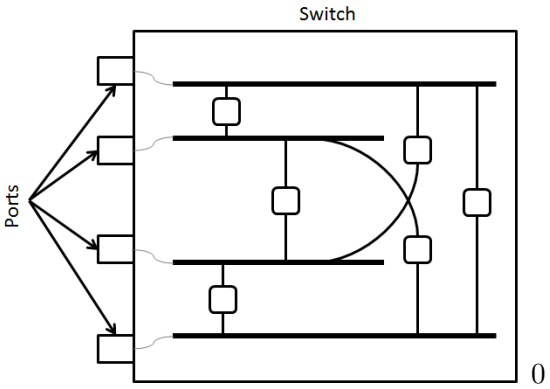


Figure 25: A *conceptual* model of a switch

2.7 Hardware addressing and Frame Type Identification

LANs provide a physical path from one computer to all others. Most communication is from a single computer, to another single computer. Regular addressing allows the source computer to identify the destination computer, and *hardware* addressing prevents any unnecessary messages being seen by a computer [11]. Each computer is assigned a unique numerical address on the network, known as a hardware address (or physical address). Each frame that is transmitted will include both a destination address and a source address. The network hardware decodes each frame and uses the destination address to determine whether or not to accept it. The source address is used to make generation of replies simpler. There are three main types of hardware addresses:

- Static
  The manufacturer assigns the hardware address, and it never changes. This is easier for user (as they are not required to set anything up), and means computers can easily move between networks.

- Dynamic
  The address is generated whenever the station boots up. This is easier for the vendor, and address can be smaller. Unfortunately, this can cause conflicts to arise

- Configurable
  The user can set the address themselves. This allows for smaller, permanent addresses.

Broadcasting
There is also another address that is present in networks, which is the broadcast address. This is a special reserved address that can be used to transmits a frame to all stations on a network.

Multicasting
A restricted form of broadcasting, called *multicasting*, is also available. This has the capability to broadcast, without wasting CPU resources on all computers (for instance, it allows a select few computers to be transmitted to). This extends the addressing scheme by reserving some addresses for multicasts, and extends the network interface to accept an additional set of addresses.

Promiscuous Mode
As mentioned earlier, there is a special mode that a network hardware can take on, called *promiscuous mode*. This is where the hardware will accept all packets sent through the network, whether they are the destination or not. This is useful for network analysers, but can have servere security implications.

Identifying Packet Contents
In order to process some packets, computers must know of what type they are. There are two ways of doing this: *explicit frame types*, which are specified by network hardware designing to determine how type information is included in the frame; and *implicit frame types*, where the sending and receiving computers must agree how types are specified.

As far as the actual format[6] of a frame, they typically include some header, and some data area. The header has a fixed size, and the data area varies between some minimum and some maximum. For example (and as displayed in figure 26), the Ethernet frame format uses a 6 byte[7] destination address, a 6 byte source address, with 2 bytes used for the frame type. It also specifies that the data be between 46 and 1500 bytes long (with a 4 byte CRC).

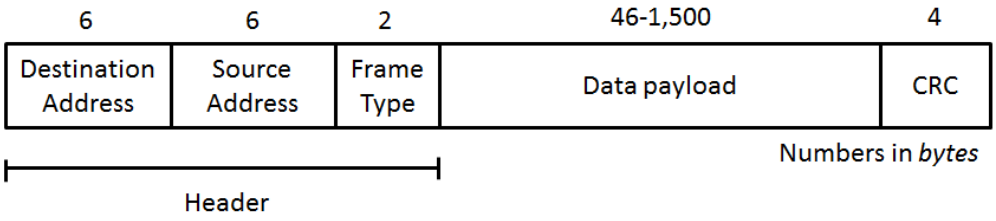


Figure 26: Specification of the Ethernet frame format

2.8 Wide area networks (WANs)

Wide Area Networks were developed before LANs emerged (before personal computers were available), and were tasked with connecting a set of sites. This wide scale connectivity is achieved through many connected switches, each allowing for local computers to be connected to other switches - this concept is demonstrated in figure 27.

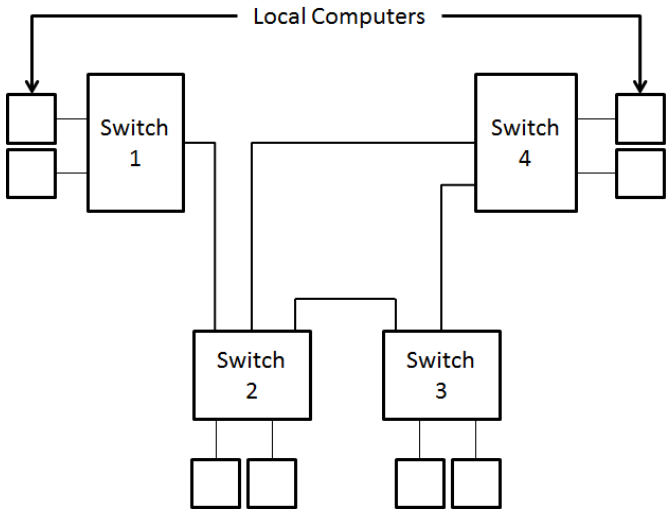
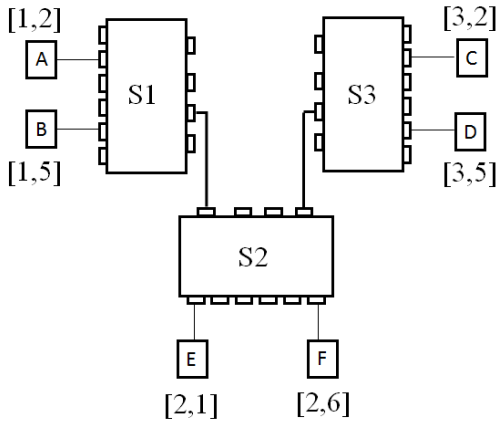


Figure 27: Example WAN, showing the connection of local computers to other switches

[6]The way a packet is organised, including details such as size and meaning of individual fields.
[7]*Byte*, not bit. Remember, one byte is equal to eight bits

The communication between the computers on different sites is dealt with by *next hop forwarding*. A *hop* is one step through the network, to either a computer or another switch. The next hop for each packet is determined by its destination address (the source address is irrelevant, which is known as *source independence*), and a listing of all *next hops* is present in a *routing table*. The addresses for the computers is comprised one part by the switch they are on, and one part a unique identification within this switch. An example routing table, demonstrating how these addresses are comprised, is displayed in figure 28



Routing table from S2

Destination	Next Hop
[1,2]	interface 1
[1,5]	interface 1
[3,2]	interface 3
[3,5]	interface 3
[2,1]	computer E
[2,6]	computer F

Figure 28: Example routing table

The hierarchical addressing means routing tables can be condensed, and look up is more efficient [22]. For instance, instead of having [1,2] linking to interface 1 *and* [1,5] linking to interface 1 in our routing table, we could just have a single entry of [1,x] linking to interface 1. The routing in a WAN must guarantee two things: Universal routing, there must be a next hop for every possible destination; and Optimal Routes, the next hop must point to the shortest path to the destination. It is not feasible to manually compute the routing table for a large network; but there are algorithms to do this for us. The computation of a routing table can be either *static*, an unchanging table that is generated when the switch boots, or *dynamic*, which changes over the time as conditions within the network change.

It is easy to conceptualise WANs as a graph of interconnected nodes, where the weights of the connections are the physical distance between the nodes. Any shortest path algorithm can then be applied to this tree; however the two we will cover are *Dijkstra's Algorithm* and *Distance Vector Routing*.

Dijkstra's Algorithm

Given

A graph with non-negative weights assigned to each edge and a designated source node

Compute

The shortest distance from the source node to each other node, and a next-hop routing table

Method

Using three data structures: D, an array that holds the current distance to each node; R, an array that holds the next hop destination for each node; and S, a linked list of all remaining nodes, iteratively select the node, u, from S, that has the smallest current distance from the source, examine each neighbour of u and, if the distance to any neighbours is less via u that currently recorded for that neighbour, update it. The full source code of the algorithm can be found in appendix A.

A good example of how to apply Dijkstra's Algorithm can be found on the Pearsons Schools and Colleges website [7] (starting on page 25).

The second algorithm, *Distance Vector Routing*, periodically computes new local tables for switches, which is then distributed among its neighbours.

Distance Vector Routing
Given

A local routing table, an incoming route message, and weights for each link that connects to another switch

Compute

An updated routing table

Method

A "distance" field should be maintained with each routing table entry, this routing table is initialised with a single entry that has destination of the local packet switch, an unused next hop, and a distance of 0. The algorithm waits for a routing message to be received over the network, and uses this to recompute its own weights and a result. The full source code of the algorithm can be found in appendix B.

2.9 Protocols and Layering

The final element of this section of the module concerns protocols and layering, and their importance. Basic communication hardware can transfer bits from one place to another, and communication software provides a high level interface for application programmers (so they do not have to deal directly with hardware) [20]. A set of rules that governs the exchanging of messages over a network is called a *network protocol*. Instead of extremely large, single, protocols, we implement *protocol suites*, which are sets of multiple protocols that interact with one another. This makes designing, testing, extending, and updating the protocols much simpler.

The most common approach to implementing these suites is called *layering*. Each layer deals exclusive with those layers one above, and one below themselves. This means each layer can handle different levels of abstraction, and there may be several alternatives for each layer. One example of a layer model is the *Open Systems Interconnection Seven Layer Model* - OSI[8] 7 Layer Model for short. This is an old model (20 years old now), is too complex, and is woefully out of date, but provides useful terminology and a good example of layer - this model is displayed in figure 29.

7 - Application
6 - Presentation
5 - Session
4 - Transport
3 - Network
2 - Data Link
1 - Physical

Figure 29: The OSI 7 Layer Model (Developed by ISO)

As data traverses through the different layers of the model, a new header is appended, to give layer-specific information. This packet is then removed again as it traverses back up the layers, being read and removed by the corresponding layer (with the exception of the Physical Layer, which will very rarely append a head). This can be summarised to: *Layer N software on the destination computer will receive exactly the message sent by layer N software on the sending computer.*

[8]The OSI model should not be confused with the company that developed this model, who are known as ISO

There are many common networking issues; of which we have discussed the important ones below

- Sequencing for out-of-order delivery
  Sometimes, networks can deliver packets that are not in the correct order - this is solved through *sequencing*. Each packet is assigned a sequence number, and the receiver can use this number to determine the order the packets should be in, and can arrange them accordingly.

- Sequencing to eliminate duplication of packets
  This sequencing method can also be used to eliminate duplication of packets - when two packets have the same sequence number, one can be deleted.

- Re-transmitting lost packets
  Due to transmission errors, it is sometimes possible that packets can be lost. One approach to produce more reliable transmissions involve "positive acknowledgements". The sender transmits, and then starts a timer - which counts to some limit, called a *time-out*. When the receiver receives, it will send an acknowledge and the timer will cease. If the receiver however, does not receive the packet, it will be transmitted again when the timer hits the time-out.

- Avoiding replay caused by excessive delay
  Duplicate packets may appear later in a session for one reason or another. It can then be confused with a packet from a newer session that has the same sequence number. To solve this, packets include both their sequence number, and some session identifier.

- Flow control to prevent data overrun
  This is the problem where senders send faster than the receiver can receive. A simple solution to this would be to wait for an acknowledgement on each packet's successful delivery before sending the next. However, this can be wasteful of bandwidth, a solution to this would be a sliding window protocol; discussed below.

Sliding Window Protocols
In the sliding window protocol, the sender and receiver are both programmed to use a fixed window size, which is the maximum amount of data that can be sent before an acknowledge arrives. The "window" term represents the logical boundary of the total number of packets yet to be acknowledged by the receiver. The sender begins with the data to be sent, extracts data to fill x packets, and transmits a copy of each packet (a copy is retained for retransmission purposes). When the packets are received in order, the receiver sends back an acknowledge. Only after this has then been received by the sender, will the packets be discarded from its memory, and the slider moved along. The sliding window method ensures that traffic congestion on the network is avoided. By placing a limit on the number of packets that can be transmitted or received at any given time, a sliding window protocols allows an unlimited number of packets to be communicated using a fixed-size sequence number. The reason that a sliding window protocol helps prevent "stop and go" performance, is that, the protocol does not wait for the acknowledge of each packet individually before moving on (as this is very wasteful of bandwidth), and instead looks at the window. Figure 30 shows how the use of a sliding window protocol can drastically decrease transmission time.

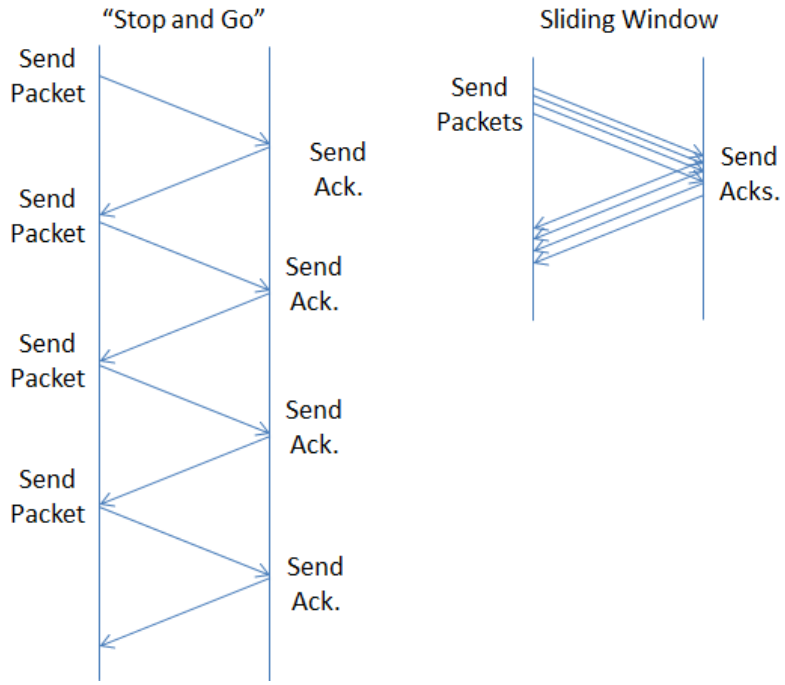


Figure 30: Diagram showing the effectiveness of a sliding window protocol against a "stop and go" protocol

Congestion

Congestion occurs when their are bottle necks in the network, or too much traffic is being sent. For an example, imaging two switches connected by a 1Gbps connection, each with two computers attached with the same connection speed. If both computers on one of the switches attempt to transmit at the same time, this will be 2Gbps of data, attempting to traverse a connection that only allows upto 1Gbps - some of the data will have to wait, which can sometimes cause packets to be lost. It is important to detect congestion, so that it can be dealt with. Detection is handled by arranging for some intermediate system that informs senders when congestion occurs, and uses the delay length or number of packets lost as an estimate of how much congestion there is. Once congestion is detectable, it can be prevented by reducing the rate at which packets are being transmitted.

3 Internetworking

Large organisations will generally use several different networking technologies, and inter-organisational communication is an important aspect [13]. Any two computers should be able to communicate with each other, this is known as *Universal Service*, and the purpose behind this is that a user should not need to change computers to execute a specific task. But it should be noted that one does not simply wire two different network technologies together. This is why the concept of *Internetworking* was introduced. The purpose of Internetworking is to connect heterogeneous[9] networks, in order to create the illusion of a single network, known as a *virtual network*, like the one shown in figure 31.

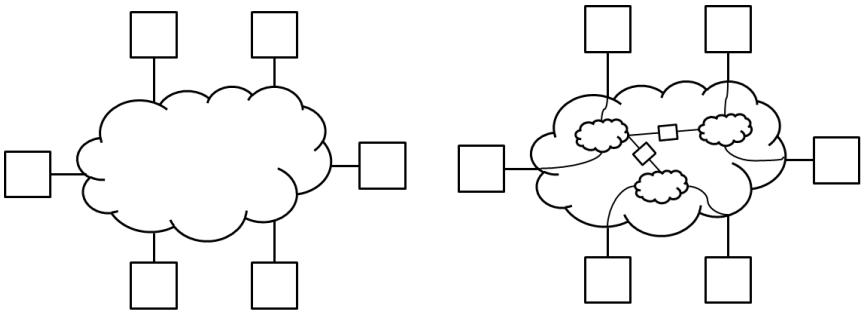


Figure 31: How an internetnetwork appears (left), against how it actually is (right)

The most prominent Internetworking protocol is *TCP/IP*. This protocol began in the 1970s, and is regulated by the Internet Engineering Task Force (IETF). Like the OSI 7 Model from before, TCP/IP is also a layered model, shown in figure 32, and will be covered in more detail in section 3.5. Remember, ATINP!

5 - Application
4 - Transport
3 - Internet
2 - Network Interface
1 - Physical

Figure 32: The TCP/IP Layer Model

In the TCP/IP protocol, a *host* computer is any system that connects to an Internet and runs applications. The TCP/IP protocol is implemented on both these host computers, and the network routers.

To allow for communications between the computers on an Internet, the need some sort of uniform addressing scheme. The most common of these (used in TCP/IP) is specified in the *Internet Protocol*. Each computer is assigned a unique 32 bit address, for identification. This address is divided into two parts: the prefix, which determines the physical network to which the host is attached (globally coordinated); and a suffix, which determines a specified host on a specific network (locally coordinated). The size of these prefixes and suffixes govern the maximum number of networks, and the maximum number of hosts on these networks. There are five different classes of addresses, with different sized prefixes and suffixes, this is specified in the first four bits of the address, and are shown in figure 33. IP addresses are 32 bits long, and sometimes can be difficult to read, so they are generally written as four decimal numbers, separated by dots (for instance, *255.10.55.192*).

[9] This is just an extortionately fancy way of saying "diverse"

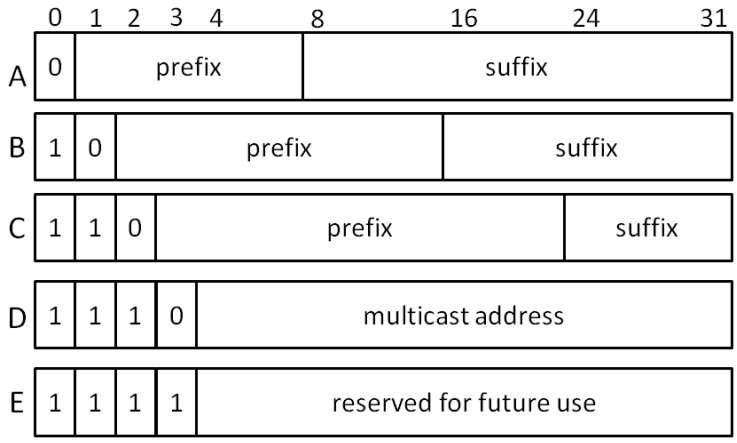


Figure 33: The five classes of IP addresses

As mentioned above, the various sizes of suffixes and prefixes affect the number of networks, and the number of hosts on these networks, this is shown in figure 35.

Class	Prefix Size	Max Networks	Suffix Size	Max Hosts Per Network
A	7	128 (2^7)	24	16,777,216 (2^{24})
B	14	16,384 (2^{14})	16	65,536 (2^{16})
C	21	2,097,152 (2^{21})	8	256 (2^8)

Figure 34: Various number of networks and hosts on these networks, as a result of prefix and suffix size

There are also some special IP addresses that you should be aware of, these are listed in figure **??**.

Prefix	Suffix	Type of Address	Purpose
all-0s	all-0s	This Computer	Used during bootstrap
network	all-0s	Network	Identifies a network
network	all-1s	Directed Broadcast	Broadcasts on a specified network
all-1s	all-1s	Limited Broadcast	Broadcasts on a local network
127	any	Loopback	Testing

Figure 35: Special IP Addresses and their purposes

3.1 Classless Addressing

In *classful* addressing, IP addresses are divided into Network and Host portions (along Octet boundaries) [6]. However, in classless addressing, the IP address is simply treated as a 32 bit stream of 0's and 1's, where the boundary between the two can fall anywhere between 0 and 31. The network portion of this address is determined by how many 1's are in the subnet mask. While this then can fall on an octet boundary, it does not have to. A subnet mask is used locally on each host connected to a network, but are never carried in IPv4 datagrams. Instead, all hosts on the same network are configured with the same mask, and share the same pattern of network bits. The main advantages of classless addressing is that addresses allocated, are of dynamic size, there are no class imbalances (meaning there is no problem if some portions of the address space are widely used and others neglected), and the multiple-level hierarchical structure allows a small number of routing entries to represent a large number of networks.

While on the subject, "Consider a network of 9 hosts. What are the suitable prefix and suffix sizes for such a network and why?" This is a highly likely question for the exam, so a thorough understanding of how to use *extended* dotted decimal notation is important. As mentioned before, classless addressing allows for a division between prefix and suffix to appear at an arbitrary boundary. In this instance, we only need 4 bits for the host suffix (as $4^2 = 16$, and $16 >= 9$). We can use a Class C address for this, and then subdivide it into 16 addresses, each with a 28 bit prefix and a 4 bit suffix. A really easy way to decide the suffix size if the lowest power of 2 that bounds the number of hosts. The reason this number is good is that it doesn't waste available addresses, but gives some room for expansion. To specify these using "extended" dotted decimal notation (*formally* known as *Classless Inter-Domain Routing* (CIDR) notation), it is actually very simple. We write a 32 Bit IP address (as long as it not one of the reserved addresses), and add a "/" to the end, followed by the size of the "mask", which is simply thirty-two minus the suffix size (in our case, 28 bits). The only constraints of this are that the first three sets of dotted notation must be the same for each, the mask must be the same for each, and finally, the fourth dotted value must be *different*. A few examples are shown in figure 36.

255.255.255.69/28
255.255.255.0/28
255.255.255.16/28
255.255.255.240/28

Figure 36: An example of the "extended" dotted decimal notation, and some examples of some addresses for a network with 9 computers

A second element to this classless identification, used in conjunction with the IPv4 address, is the subnet mask. This is another 32 bit number, that is used alongside the IPv4 address to identify the boundary between prefix and suffix. Each one (starting from the left) of the subnet mask represents the number of bits used for suffix. The remaining zeros (of which there much be atleast one) display the prefix size. So, for our example above, our subnet mask would be 255.255.255.240. To then get a broadcast address from this, we apply a logical OR operator to the complement of the subnet mask and the IPv4 Address, see figure 37 for an example. The address we are using (255.255.255.69), is just a random one from those specified above.

$$
\begin{aligned}
& 255.255.255.69 \vee \backslash(255.255.255.240) \\
= \quad & \{ \quad \text{Applying Complement} \quad \} \\
& 255.255.255.69 \vee 0.0.0.15 \\
= \quad & \{ \quad \text{Converting to Binary} \quad \} \\
& 11111111.11111111.11111111.01000101 \vee \\
& 00000000.00000000.00000000.00001111 \\
= \quad & \{ \quad \text{Applying OR} \quad \} \\
& 11111111.11111111.11111111.01001111 \\
= \quad & \{ \quad \text{Converting to decimal} \quad \} \\
& 255.255.255.79
\end{aligned}
$$

Figure 37: Example of attaining the broadcast address from a IPv4 address at its corresponding subnet mask

3.2 Addressing methods

Some different methods of broadcasting include:

- Directed Broadcast Address
  This method is used to send a copy of a packet to all hosts of a network. When a packet is sent to a directed broadcast address, it travels along the network and is then copied and transferred to each host on the network. A directed broadcast address has a suffix of all 1 bits.
- Limited Broadcast Address
  A limited broadcast refers to a broadcast on a local physical network. This is generally used during system start up that does not yet know the network number.
- Loopback Address
  The loopback address is primarily a debugging address. Instead of having to test TCP/IP code on a computer on the network, the computer can use itself can be both sender and receive (hence the name, *loop* back). The prefix of 127 is reserved for loopback.

Address Resolution
The process of translating a computer's IP address to an equivalent hardware address is known as *address resolution*, and there are three main techniques of doing this, each of which are implementations of the *Address Resolution Protocol*, and define the format of the resolution requests responses.

- Table Look up
  Routers use a routing table to determine the next destination to send the packet. This is useful with any hardware, and the protocol address is independent of the hardware address. Resolutions are produced with minimum delay, but an address change will affect all hosts.
- Closed-Form Computation
  Local hardware addresses are used for communications. These addresses must be smaller than the protocol address and, as such, is determined by the hardware address. Resolutions are still produced with minimum delay.
- Message Exchange
  Messages are sent to a specific server, and then broadcast over a network, to which only the required computer computer will then respond. The protocol addresses are, again, independent of the hardware address

  In which a message is send to a specific server, and then broadcast (only the required computer will then respond). Protocol addresses are independent of hardware address, special broadcast software is required, adds extra traffic to the network, and implementation is more difficult than other methods.

Dynamic Host Configuration Protocol
DHCP requires a special server, that is used to assign IP addresses to hosts. Newly booted machines broadcast a special "discover" packet, that the DHCP server evaluates, so that it can send back an IP address for the new machine to take on. There are two varieties of IP address in this instance: Permanent IP Addresses, which are manually assigned by an administrator; and Automatic IP Addresses, which come from a pool of addresses and are leased for some finite length of time.

3.3 IP Datagrams

Packets of data are sent across multiple physical networks and, as such, there needs to be a universal virtual packet that all routers can interpret [14]. This virtual packet is known as an *IP Datagram*.

The size of an IP Datagram is not fixed and is determined by some application. The internet protocol attempts to deliver these datagrams as best it can, but does not guarantee there will be no duplication, delay, out of order delivery, corruption of data, or datagram loss - these issues are instead dealt with at other protocol layers. The format of the IP datagram is specified in figure 38.

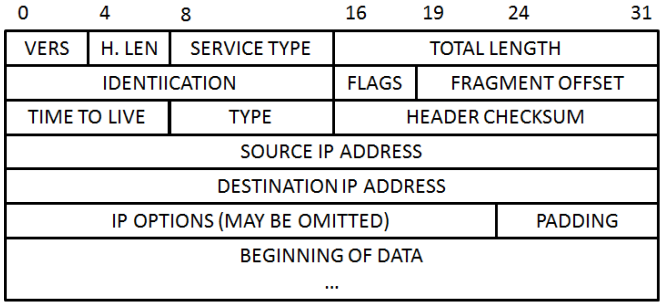


Figure 38: The format of the IP Datagram

Encapsulation
When IP datagrams are sent across a physical network, the are placed in the data area of a frame, and the frame type is set to IP; as shown in figure 39

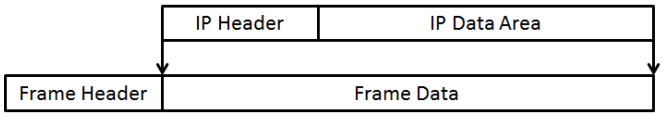


Figure 39: Encapsulation of datagrams within frames

The maximum amount of data that a frame can carry on a given network is known as the *maximum transmission unit*. As a packet passes over an Internet, it may have to copy with different MTU sizes. If a datagram is larger than one of these MTU sizes, it has to be split into several smaller datagrams. The reassembly of these datagrams in then done at the final host. It does this by using certain header fields that indicate where each datagram belongs in the sequence. If any segments of the datagram are lost, the whole thing is lost.

3.4 The Future of IP

Whilst IP has been succesfful at coping with the expansion of the Internet thus far, its limited address space is fast running out, and there are new technologies emerging that require a more reliable transfer protocol. The current IP standard is *IPv4*; but this will soon be moving to *IPv6*. This will feature 128 bit address sizes, extension headers, and support for audio and video. There will also be support for different addressing schemes, including: Unicast, Multicast, and Cluster.

- Unicast
  Corresponds to a single computer, where datagrams will be sent along the shortest route.
- Multicast
  Corresponds to a set of computers who's members can change at any time. A copy of the datagram is sent to each computer, but only one copy is actually sent across the network to the address. This is a good method for collaborate applications.
- Cluster
  Corresponds to a set of computer that share a common prefix. This method is generally used for replicating services.

When writing these new IPv6 addresses, we no longer used the dotted decimal notation, instead opting for a more compact hexadecimal colon notation (see figure 40). We combine this with *zero compression* to further reduce the space taken (figure 41).

$$
\begin{aligned}
& 105.220.136.255.255.255.255.0.0.18.128.140.10.255.255 \\
= \quad & \{ \quad \text{Applying Hexadecimal Colon Notation} \quad \} \\
& 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF
\end{aligned}
$$

Figure 40: Display of IPv6 in dotted decimal, and hexadecimal colon notations

$$
\begin{aligned}
& FF0C:0:0:0:0:0:0:B1 \\
= \quad & \{ \quad \text{Applying Zero Compression} \quad \} \\
& FF0C::B1
\end{aligned}
$$

Figure 41: Display of IPv6 in hexadecimal colon notation, and the same address after zero compression

Collaborative Virtual Environments
Multicast addresses are suited for collaborative technologies, one such technology is *Collaborative Virtual Environments*. In these environments, each client can be both an active sender, and a receiver. There may be hundreds of users within these environments, each of whom may be sending data at the same time. As a result of this, CVEs can generate large volumes of network traffic (think massively multiplayer online gaming).

3.5 Transport Control Protocol (TCP)

By itself, IP is unreliable; which is troublesome when application programmers require reliability[21]. This is why TCP, the transport control protocol, in general is used in conjunction with IP. The combination helps to establish a reliable end-to-end communication, as TCP includes the following seven main features:

1. It is connection Oriented
   Applications request a connection be established, and use this to transfer data
2. It is Point-to-Point
   There are two endpoints in the connection (there is no multicast or broadcast)
3. There is complete reliability
   There is a guarantee that data will be delivered without loss, duplication, or transmission errors
4. There is full duplex communications
   Endpoints can exchange data in both directions simultaneously
5. The connection start-up is reliable
   There is a guarantee that start up between end points is reliable and synchronised (uses 3-way handshake)
6. Shut down of connections is graceful
   There is a guarantee that all data will be delivered after endpoint shut down
7. The stream interface
   Bytes are transferred as a contiguous stream

An acronym to remember this is *PRODUGS*:

P **P**oint-to-**P**oint

R Completely **R**eliable

O Connection **O**riented

D **D**uplex Communications

U Reliable Start-**U**p

G **G**raceful shut down

S **S**tream Interface

TCP uses specific methods to ensure reliable transmission, these are:

- Adaptive Retransmission
  In this method, the sender starts a timer when it sends data across the network. If it does not receive an acknowledgement after a certain time frame, it will *time out*, and retransmit the packet. Sensible timers vary greatly on the specific Internet. TCP monitors the delay on a connection, and adapts the timer as necessary (through use of weighted average and variance over many transmissions).
- Flow Control
  Flow control is dealt with by the window mechanism (as discussed before). Each end of the connection allocates a buffer and notifies the other end of the size of this buffer (the window size). After each set of data is transmitted, the receiver sends an acknowledgement, detailing the available window size (called *window advertisement*); this window decreases when the application consumes some data. When the window size is zero, the sender stops transmitting until further notice.
- Congestion Control
  Congestion control in TCP/IP is monitored through message loss. When the first message is lost, TCP stops transmitting, and sends one small message to test how severe the congestion is. If this small message is not lost, the data size is doubled, and this is transmitted. This continues exponentially until half of the receivers window size is reached.
- 3-Way Handshake
  Special synchronisation and finish messages are used to open and close transmissions. This is helpful to ensure that all data has been sent on both ends.

3.6 Connectionless Or Connection Oriented?

In connectionless communications, no effort is made to set up a dedicated end-to-end connection between the two communicating bodies [5]. This is usually when data is transmitted in one direction, regardless of if the destination is there, or ready to accept the information. When there is no interface, and plenty of speed, these systems work well. However, in environments where there is difficulty transmitting, information may need multiple retransmissions until a complete message is received.

In connection oriented systems, conversely, an end-to-end communication is set up; generally through handshaking. This handshaking can range from being very complex, to being very simple. Connection oriented systems can *only* work in bi-directional communications; as to negotiate the connection, both sides must be able to communication.

4 Network Applications

4.1 Client - Server Paradigm

The Client-Server Paradigm is a widely used form of communication, in which the client actively initiates contact with a server, and in the mean time, the server is idle waiting for these connections [9]. The servers generally provide one specific service, and the client connects because they wish to make use of this service. Information can flow both directions when the two parties are communicating, and the most common situation is where many clients are interacting with each server. These two components communicate using a transport protocol, and are unaware of any another underlying layers.

Clients
These are general applications that perform local computation, but become clients when remote access is needed. This remote access is initiated by a user when required, and will execute for one session before returning to their local state. The client software will actively initiate connection with a server, and can access multiple services as needed - but only one at a time.

Servers
Special purpose programs (generally running on a fairly powerful, special purpose machine), that provide one specific service and can handle communications between multiple clients at one time. Unlike clients, the server software is always running (after the system boots), and runs through many sessions. The server waits passively for contact from various kinds of client, but still only provides one service to each. It is not uncommon for one server machine to run multiple different servers on it.

The transport protocol assigns a unique identifier to each service provided. When the server boots, it registers its own service with the transport protocol to receive a unique ID. When a client connects to a server, it specifies the ID of the service it wishes to use. These IDs are called *protocol port numbers*.

Concurrency and Servers
Concurrent[10] servers offer support to multiple clients at the same time; through the use of *multiple threading*. There is a core part of the server that accepts new request and dynamically creates new servers as separate threads to handle them. This means that at any point, the server will be running $n+1$ threads, where n is the number of clients. Each of these threads deals with a single client's requests, and TCP uses a combination of destination port and source port to identify the correct thread.

Complex Client-Server Interactions
A client is not restricted to accessing multiple services (although it *is* restricted to a single service at any one time). A client is also not restricted to using a single server for one service - it can access the same service across multiple servers. It's also worth noting that servers themselves can become clients of other servers, but this can lead to potential circular dependencies.

[10] God help us all

4.2 Distributed Programming

Distributed Programming is an extension and generalisation of the client server paradigm. This allows for greater transparency for programmers, and allows access for tools such as remote procedure calls (execution of a procedure in another address space), and distributed objects and components. There are several implementations of distributed programming: traders, factories, and peer to peer communications.

Traders
Traders can be thought of the middle men of the communication. As shown in figure 42, the interaction begins by the service detailing to the trader, what service they provide. The client then interacts with the trader to request this service, after which the client can then actually use the service.

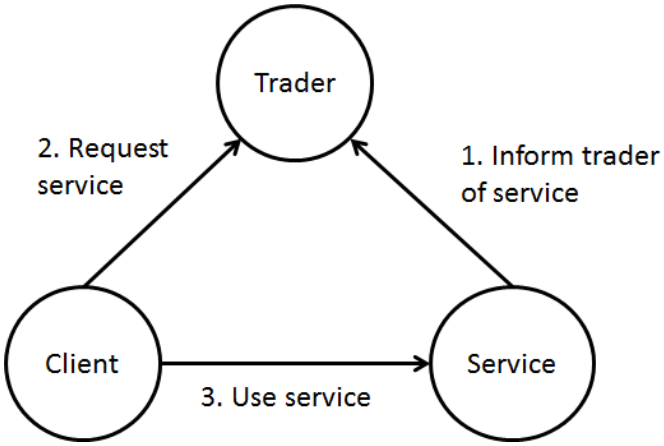


Figure 42: Conceptual model of the trader interaction

Factories
Factories add an extra element to the trader interaction. In this case, the client is the element that begins interactions. The client requests a specific service, which the trader will then talk to the factory about. The factory will then create this service, and make it available on the service part of the server. The client can then access and use this service (shown in figure 43).

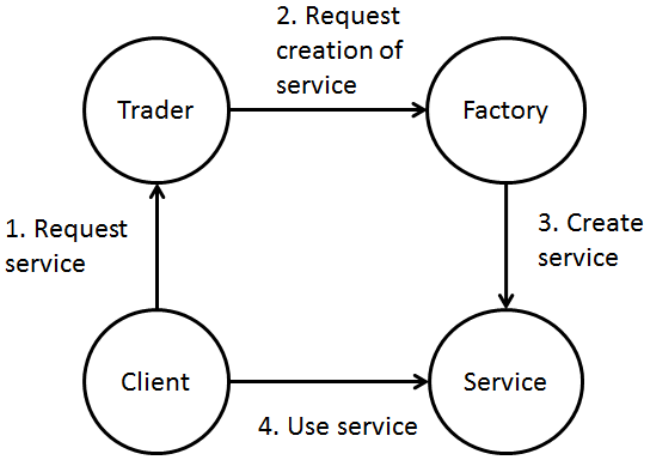


Figure 43: Conceptual model of the factory interaction

Peer to Peer
When multiple clients are connecting to a single server, the connection can become a bottleneck. To avoid this, peer to peer communications (p2p) do not have one centralised server. Instead, every computer on the network is considered the server. This means that, when a computer requires a service from this network, it receives $\frac{1}{N}$ of the service from each of the N computers. This means the load on any one computer is severely reduced, shown in figure 44.

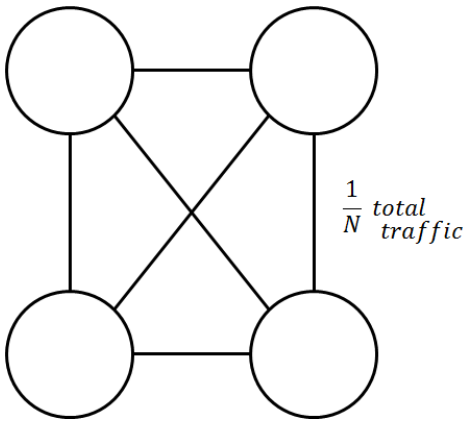


Figure 44: Conceptual model of peer to peer interaction

4.3 The Socket Interface

This Socket API is the de facto standard for TCP/IP communications. Socket interaction differs from conventional I/O systems because applications must specify many details before interaction can begin. These details include transport protocol to be used, protocol address, and whether or not the application will act as a client or server. The application requests that the operating system create a socket, which is identified by a *descriptor* - a small integer. Communications can then take place, and any reading or writing of data requires the descriptor as argument. When the communications are finished, the socket is closed.

We will now go through the seven main socket functions, commonly referred to as the Berkeley socket functions (BSD for short). The order that these functions are called in shown in figure 45.

- *socket(protofamily, type, protocol) returns descriptor*
  Function that creates a socket and assigns it to a descriptor; *protofamily* specifies protocol family, *type* specifies the communication type, and *protocol* specifies a specific protocol. A socket can be closed with a call to the *close(socket)* function.
- *bind(socket, localaddr, addrlen)*
  Associates a socket with a protocol port number; *socket* is the socket's descriptor, *localaddr* is a structure containing a local address, and *addrlen* is the length of this address.
- *listen(socket, queuesize)*
  A function used by the server to put a socket (*socket*) into passive mode; *queuesize* is the length of the request queue.
- *accept(socket, caddress, caddresslen) returns socket*
  A function used by the server to accept a new connection from a client; *socket* is the socket to be accepted, *caddress* is the client address, and *caddresslen* is the length of this address.
- *connect(socket, saddress, saddresslen)*
  A function used by the client to connect to a server; *socket* is the local socket for the client, *saddress* is the server's address, and protocol port number, and *saddresslen* is the length of this address.

- send(socket, data, length, flags)
  Function that sends data; *socket* is a socket to send to, *data* is a *pointer* to the data, *length* is the length of the data, and *flags* are used to enabled debugging options. (*sendto* and *sendmsg* are used with connectionless communications).

- recv(socket, buffer, length, flags)
  Function used to receive data; *socket* is a connected socket, *buffer* is a pointer to allocated memory for the data to be stored in, *length* is the size of this buffer, and *flags* is used to control additional details (*recvfrom* and *recv*

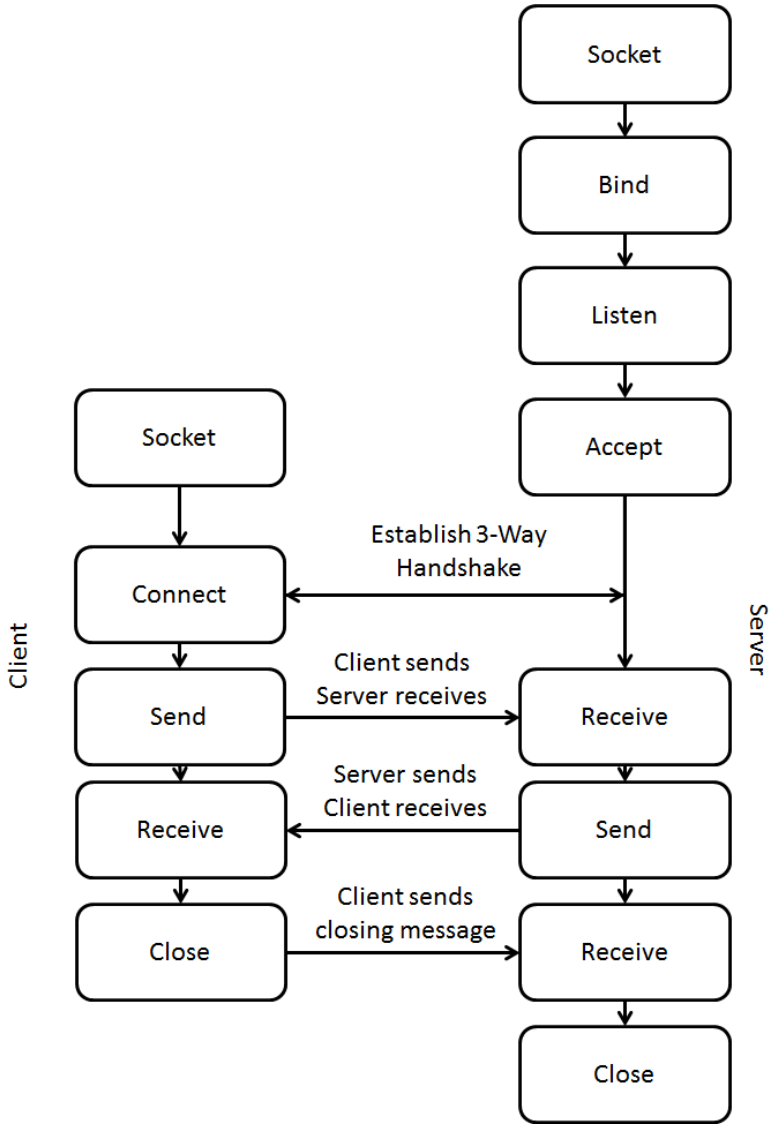


Figure 45: The flow of function calls in both the server and client during socket interaction

Some other functions that are also available include: *getpeername*, which gets the complete address of an initiating client; *gethostbyname*, which can be used to get information about a host; and *gethostbyaddr*, which maps an IP address back to a host name.

A good way to remember each of these functions is the following acronym: "Silly Bastards Lack Accurate Reading Comprehension Skills" (SBL ARCS for short). These stand for Socket, Bind, Listen, Accept, Receive, Close and Send. If you flip Close and Send, this is the exact order that a server would interact with these functions.

The secure socket layer (SSL)
Briefly, SSL is a cryptographic protocol that provides security for communications over the Internet [2]. Several versions of the protocol are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). It generally sits between the application layer, and the transport layer. In TCP/IP, SSL encrypts the data at a lower sublayer of its application layer, and relies on the properties guaranteed by TCP. After a connection is initiate by TCP, the SSL handshake is started by the client. The client specifies which version of SSL it is running, what ciphersuites it wishes to use, and what compression methods should be used. The server then checks what the highest SSL version supported by them both is, and picks a ciphersuite and compression method from one of the client's options. After this, the server sends a certificate, which must be trusted by the client itself, or a party the client trusts. Having verified the certificate and being certain the server is truly who it claims to be, a key is exchanged - the format of which is specified in the ciphersuite. Both parties compute this key, and at this point, the client sends an encrypted message to the server, telling it that all further communications will be encrypted also. If the server successfully receives and decodes this message, the handshake is complete, and the two parties can communicate securely.

4.4 Network Security

In computer security there are two main types of threats, passive and active [18]. Passive attackers just listen to data being transmitted over the network (for things such as eavesdropping, wire trapping, sniffing). It can be used in: traffic analysis, to identify communication patterns; linkability, to identify messages coming from the same source (who is talking to who); and in mobile services, to track the location of the user. Active attackers modify messages, inserts new messages, and even corrupts network information. There are two main methods for this: Spoofing, where messages come from a forged sender; or Squatting, where the attacker claims to be the victim's location (phishing).

One example of an attack is called *TCP SYN Flooding*. Which exploits the 3-Way Handshake to flood a server with synchronisation requests. The attacker begins by intiating a 3-Way handshake, but immediately breaks it, so that is never finishes. While the victim is busy with the time out counter, the attacker floods this system with a large number of synchronisation requests. The victim then reaches its half-open connection limit, and service to other connections is denied.

4.5 IP Security & IPSec

IP is connectionless and stateless (each datagram is treated as a separate entity) and provides best-effort service, but does not guarantee delivery of packets, has no mechanism for maintaining delivery and (in IPv4 atleast) has no security protection. IPv6 implements a mandatory IPSec, which contains two major security components: IP Authentication Header, and IP Encapsulation Payload (but does not contain a mechanism to prevent traffic analysis).
The IP Authentication Header protects the integrity and authentication of IP packets, but does not protect confidentially. Everyone can read the TCP packet, but if they change it, it will be detected. IP Encapsulation Payload provides confidentiality, data origin authentication, replay protection, and some traffic flow confidentiality; all of which is achieved by encrypting the payload.

There are then two ways of transmitting this payload: Transport Mode, where the frame is encapsulated and encrypted, providing an end to end protection of packets (but requires the end host to be aware of IPSec); and Tunnel Mode, where the entire datagram is treated as a new payload (which can be performed at security gateways - think IP within IP), and provides traffic flow confidentiality (with the additional bonus that the end host need not be IPSec aware).

IPSec services use encryption, but are not tied to one particular key management system, which prevents them using a key management system that is later found to have security flaws. It provides transparent security for everyone using IP, without changing the interface of IP, and provides host-to-host security without an additional overhead.

4.6 DNS Security Issues

The most prevalent problem with DNS is that attackers can corrupt DNS information, and thus redirect users to a fake site, or even make sites seem unavailable (DoS Attack). This becomes worse when the corruption is propagated between DNS servers. DNSEC, a secure DNS server, is underway.

4.7 Firewalls

A firewall is, "*A network device controlling traffic between two parts of a network*". These are generally installed between a LAN and an Internet, between different LANs, or on individual hosts. These are only effective if *all* traffic goes through it. The job of the firewall is to protect networks from external parties accessing services that should only be available internally.

Packet Filters
Packet filters specify which packets are allowed, and which are dropped. This is determined using the source address, destination address, TCP and UDP numbers. It is usable on both inbound and outbound packets, and can be implemented in a router examining packet headers. Unfortunately, this method only provides a few crude rules, which cannot be dynamically defined, and some protocols are difficult to handle (even some common ones).

Application-Level Proxies
These are typically run on a hardened PC, and provide close control over content, offering a high level of security. Unfortunately, this comes with a large overhead per connection, are more expensive than packet filters, require complex configuration, and a separate proxy is required for each service needing protection.

There are two policies that firewalls can implement: Permissive, where everything is allowed except dangerous services - the problem with this being that it is easy to forget something; or Restrictive, where nothing is allowed except designated services, which is more secure but can lead to the problem of blocking something that is important.

It is important to note that the location of a firewall is important to its effectiveness. There are also some inherent issues with firewalls, these being: there is no protection against insider threats, and encrypted traffic cannot be examined.

4.8 Intrusion Detection Systems

Many of the methods discussed above are good, but it is impossible to prevent all attacks (DoS attacks, insider attacks, already happening attacks), which is why Intrusion Detection Systems were created. These consist of a number of sensors, which collect various data. This data can then be analysed, and intrusion report, so that certain actions can be triggered. Two main methods of detection are:

- Misuse Detection
  These systems look for attack signatures, which are certain patterns of network traffic, for instance a large number of failed log on attempts. This method is only as good as it's database of attack signatures, as someone must manually update the IDS database every time a new attack is recognised and a signature need be produced.

- Anomaly Detection
  In these systems, statistical techniques are used to identify strange behaviours. The first "normal behaviour" is measured as the baseline, and any actions that deviates from the baseline and exceeds a certain threshold, and alarm is issued. This system has the potential to detect novel attacks, but (as the name suggests) can only detect anomalies. Anomalies, however, are not always attacks, and attacks are not always anomalies. It is possible to register an alarm for an action that is not an attack (False Positive), and detect an attack as a normal activity (False Negative).

The most effective IDS use both a network based system, where attack signatures are garnished from the network traffic, and a host based system, where attack signatures are recognised from system activity.

4.9 Vulnerability Assessment and Honeypots

Vulnerability Assessment is a process in which the security state of a network or host is examined, for instance open ports and package information. This allows potential flaws in the system to be identified and, following this, rectified.

Honeypots are resources that track attackers, and gather evidence about their activities. They are designed to mimic real systems, so the attacker will not suspect anything when connecting.

4.10 Example Network Applications

There are many different types of network applications, including end-user applications, such as email, file transfer, and the World Wide Web, and other more broad applications, called supporting service, that encompass things such as domain name registration, encryption and billing [10]. To help identify and interact with these application, we can give computers symbolic textual names to identify themselves, instead of numeric IP addresses. These names are sequences of alphanumeric strings separated by full stops, structured hierarchically, with the most significant part on the right. In figure 46, we see an example DNS hierarchy. The circled address is not what you would at first assume it to be; instead of being *root.edu.cmu.cs.cmcl*, we read it the opposite way and without the root segment, this produces *cmcl.cs.cmu.edu*.

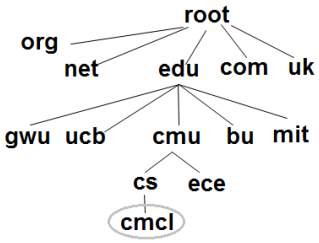


Figure 46: DNS hierarchy of some domains

Management of domain names is done by the Internet authority, and organisations are free to register top levels domains from them. The organisation can then manage their own name space, and even further develop the domain into sub-domains, for subsections of the organisation. Figure 47 gives a quick list of some well known top level domains.

Abbreviation	Meaning
com	Commercial Organisation
edu	Educational Institution
gov	Government Organisation
mil	Military Group
net	Network Support Centre
org	Misc. Organisation
info	General Informations
biz	Businesses
name	Individuals

Figure 47: A list of common top level domains

The Domain Name System (DNS) is used by applications to map these symbolic names to the actual numeric address they represent. Each entry in the DNS database contains three items: a domain name, a record type, and a value. Record types can have one of four different values: Address Type (represented by *A*, with is the IP address of a given domain), Canonical Name (represented by *CNAME*, which is any alternative names a given domain has), Mail Exchanger (represented as *MX*, which is the Mail Exchanger of a given domain), and Name Server (represented as *NS*, detailing the Name Server of a given domain).

The DNS database is distributed among many servers. When a machine requests an address, it requests a local DNS machine, which then contacts its own local servers if necessary. Root servers are authorities for top-level domains, and other servers are authorities for a part of the hierarchy, holding references to servers both lower down, and at the root. Figure 48 shows different ways domains can be split, and their corresponding names.

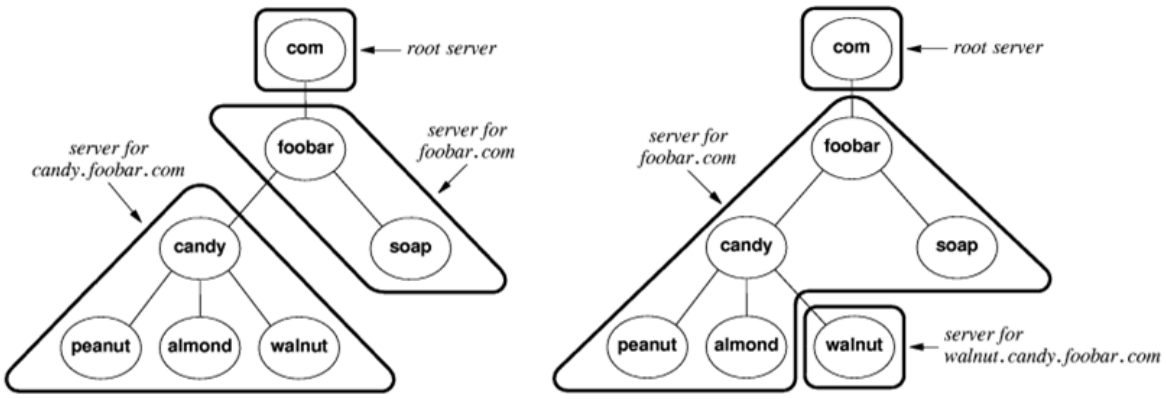


Figure 48: Different ways domains can be split

When client-server interaction is considered, the client requests a recursive resolution of domain names, and a server requests an iterative resolution. The client requests a DNS query to the DNS server asking it for the best information it has (without requesting any information from any other DNS servers). If the server does not know the answer to a DNS query, it will ask local servers whether they do or not. This continues recursively until a server with the information is found, and the answer to the query is returned. Two methods to increase the performance of this resolution is replication of servers (especially root servers), and caching, which exploits temporal locality of reference.

The effect of caching DNS response is that repeated translations have much quicker responses, and other queries may reuse some parts of lookups that have already taken place. This extends to *negative queries*, which are erroneous queries, cached so that past mistakes are not repeated. The cached data periodically times out, which is governed by the lifetime (TTL) of the data, which is set by its owner.

5 In Closing

The exam itself will be 2 hours long, with a total of 5 questions, that you are expected to answer 3 of. Each question is worth 25 points, making the maximum number of points for any exam entry, 75. This means that each mark in the exam is worth 1.33% of your mark for G52CCN. There will be four topics covered in the exam: data transmission, packet transmission, internetworking, and applications (conveniently the four topics covered in this revision guide). These are generally split across the five questions like this:

1. Data Transmissions/Local Asynchronous Communication
2. Local Area Networks/Transmission Coordination
3. Wide Area Networks
4. The Internet and TCP/IP
5. Applications and Security

In general, the marks of question 5 are good, but all other questions have had mixed results.

Some common mistakes to avoid are: providing incomplete answers, for instance giving definitions but not comparisons; providing datagrams or formulae without explanations; and providing vague answers.

References

[1] http://ironbark.xtelco.com.au/subjects/DC/lectures/7/fig_2010_07_04.jpg.

[2] http://tools.ietf.org/html/rfc5246.

[3] http://topperchoice.com/wp-content/uploads/2013/03/Introduction-To-Amplitude-Modulation-.jpg.

[4] http://www.eis.mdx.ac.uk/staffpages/yuanluo/teaching/ccm3032/Seminar-solution-6.pdf.

[5] http://www.inetdaemon.com/tutorials/basic_concepts/communication/connection-oriented_vs_connectionless.shtml.

[6] http://www.inetdaemon.com/tutorials/internet/ip/addresses/classless.shtml.

[7] http://www.pearsonschoolsandfecolleges.co.uk/Secondary/Mathematics/16plus/AdvancingMathsForAQA2ndEdition/Samples/SampleMaterial/Chp-02%20023-043.pdf.

[8] Douglas E Comer. *Computer networks and internets*. Prentice Hall Press, 2008.

[9] Milena Radenkovic. Client-server interaction, the socket api. 2013.

[10] Milena Radenkovic. Example applications. 2013.

[11] Milena Radenkovic. Extending lans. 2013.

[12] Milena Radenkovic. Hardware addressing and frame types. 2013.

[13] Milena Radenkovic. Internetworking concepts and ip addressing. 2013.

[14] Milena Radenkovic. Ip datagrams, future ip. 2013.

[15] Milena Radenkovic. Local area networks. 2013.

[16] Milena Radenkovic. Local asynchronous communication and rs-232. 2013.

[17] Milena Radenkovic. Long distance communication (carriers and modems). 2013.

[18] Milena Radenkovic. Network security. 2013.

[19] Milena Radenkovic. Packets, frames and error detection. 2013.

[20] Milena Radenkovic. Protocols and layering. 2013.

[21] Milena Radenkovic. Transport control protocol (tcp). 2013.

[22] Milena Radenkovic. Wan technologies and routing. 2013.

[23] Andrew S Tanenbaum. Computer networks. *Prentice H all PTR (ECS Professional)*, 1(99):6, 1988.

A Dijkstra's Algorithm

```
while ( set S is not empty ) {
        choose a node u from S such that D[u] is minimum
        if ( D[u] is infinity ) {
                        error: no path exists to nodes in S; quit
        }
        delete u from set S
        for each node v such that (u, v) is an edge {
                if v still in S {
                        c = D[u] + weight (u,v);
                        if c < D[v] {
                                R[v] = R[u]
                                D[v] = c;
                        }
                }
        }
}
```

B Distance Vector Routing

```
Repeat forever {
        wait for the next routing message to arrive over the network;
        let the sender be switch N
        for each entry in the message {
                let V be the destination in the entry and D the distance
                compute C as D plus the weight of the link over which
                the message arrived
                        examine and update the local routing table {
                        if ( no route exists to V ) {
                                add an entry to the local table for V
                                with next hop N and distance C
                        } else  if ( a route exists with next hop N ) {
                                replace existing distance with C
                        } else if (route exists with distance > C )
                                change next hop to N and distance to C
                        }
        }
}
```