UNIVERSITY OF LONDON | INTERNATIONAL PROGRAMMES

Data communications and enterprise networking Volume 1

P. Tarr

CO2222

2005

Undergraduate study in
Computing and related programmes

This is an extract from a subject guide for an undergraduate course offered as part of the University of London International Programmes in Computing. Materials for these programmes are developed by academics at Goldsmiths.
For more information, see: www.londoninternational.ac.uk

Goldsmiths
UNIVERSITY OF LONDON

This guide was prepared for the University of London International Programmes by:

P. Tarr

This is one of a series of subject guides published by the University. We regret that due to pressure of work the author is unable to enter into any correspondence relating to, or arising from, the guide. If you have any comments on this subject guide, favourable or unfavourable, please use the form at the back of this guide.

University of London International Programmes
Publications Office
32 Russell Square
London WC1B 5DN
United Kingdom

www.londoninternational.ac.uk

Published by: University of London

Contents

Introduction

1.1 Aims and objectives

The CIS222 module aims to provide you with a good grounding in data communications and enterprise networking. It is a level two course, building on material taught in level one modules. The course concentrates on breadth of understanding rather than depth, and attempts to cover a wide range of networking topics by means of a structured high-level approach. If you desire a deeper understanding of any of the topics covered in this subject guide, you should refer to the recommended texts.

Networks have become increasingly important in recent years. Hardly any applications are now stand-alone. Most applications communicate over a network, either between a client PC and a server in the same building over a Local Area Network or between computers in different cities, countries or continents over a Wide Area Network. The study of networks and applications that make use of networks is now an important part of any undergraduate computer science programme. An understanding of how networks operate will be useful in any computing career you may choose to follow.

The main objective is for you to build a logical framework in which networking topics can be studied, analysed and understood. This will stand you in good stead if you subsequently work in the computer or networking industry and have to design, develop or manage systems that make use of networks. This subject, like many in Computer Science, develops rapidly and new network protocols and technologies are emerging all the time. If you have built a framework which you then can use to analyse and understand these new developments, this will ultimately be of more use than a detailed understanding of current technologies.

A further objective is to help you be able to make informed choices amongst the many competing technologies that are available in the marketplace, if in a subsequent career you are required to make such choices.

The CIS222 module replaces the CIS208 module entitled **Telecommunications and Computer Communications**. Some of the material in the CIS208 module has become very dated, as new technologies have emerged and data transmission speeds have increased way beyond what was imagined possible ten years ago. Certain promising technologies which were considered important in the CIS208 module have also failed to come into widespread use, mainly as a result of the phenomenal success of the Internet which did not have a very high profile in CIS208. As a result of this, the new CIS222 module unlike CIS208 is very much focused on the Internet and its protocols.

1.2 Learning outcomes

On completion of this course and the relevant reading, you should be able to:

- explain the operation of different data communications protocols and their roles within layered network architectures
- explain the need for standards and outline the roles of the various standards bodies responsible for data communications standardisation

- describe the advantages and disadvantages of different network topologies and technologies
- demonstrate skills in using simple network management tools such as ping, traceroute and netstat and in using spreadsheets to analyse and display results.

1.3 Course outline

The CIS222 module consists of two distinct parts which correspond to the two volumes of the subject guide. The first part of the course is called 'Data Communications' and is an introduction to the terms, concepts and network architectures required to understand how data is transmitted through communications channels and networks. This, of necessity, requires a technical approach and some knowledge of the physics of transmission as well as the study of network architectures and protocols. The second part of the module is called 'Enterprise Networking' and is more concerned with the design and management of networks used by businesses and other large organisations. It therefore concentrates more on the business and management issues that arise.

This volume of the subject guide can be regarded as a comparative technical study of layered network protocols[1]. This volume contains eight chapters. This chapter will introduce the subject. The second chapter will introduce the terms and concepts that are used to describe data communications systems and the third chapter will describe network architectures and reference models. The next five chapters will examine protocols in the five layers of the hybrid reference model. There will be chapters on the application, transport, network, data link and physical layers. This approach gives equal status to each layer and ensures a consistent structure to each chapter. Each chapter has sections on services, interfaces, functions and protocols.

[1] *The rules which are used to encode and transmit data across networks and the technology that employ these protocols.*

In this part of the subject, we will follow an unconventional approach. Material will be presented in a top-down fashion. Most courses in the subject start with the physics of data transmission and build the other communications layers progressively on top of this foundation. Eventually, after all the other layers have been studied, the application layer is then examined. While this is a natural approach for electrical engineering students who will have a good prior knowledge of physics, it is less satisfactory for many computer science students, some of whom have very little knowledge of physics but virtually all of whom will have a good knowledge of network applications such as email and the world wide web. The top-down approach therefore starts with applications with which you are familiar, and then examines progressively the other communication layers, ending with the physical layer which many students find the most difficult. The main advantage of this top-down approach is that you will see the relevance of the material in the context of your existing knowledge and will not be put off the subject at the initial stage because you find the physics difficult and cannot see its relevance.

This subject guide will contain practical activities that can be carried out using simple network diagnostic tools like ping and traceroute, which are supplied with operating systems such as Windows, MacOS and Linux/Unix. These activities can be carried out in a laboratory or at home. Knowledge of how to use these simple tools will enable you to investigate network problems on your own.

1.4 How to use this subject guide

This subject guide is intended to provide a logical structure in which to study the subject of Enterprise Networking. It is not intended to replace your need to read around the subject to improve your understanding. You may wish to follow some of the further reading referenced at the start of each chapter. An alternative or supplement to the further reading is to follow the links on the course web site[2] to study the topics in each chapter.

[2] *http://doc.gold.ac.uk/~mas01pt/cis222/usefullinks.htm.*

How to best use this subject guide will depend on your personal approach to study and whether you are studying on your own or at an institution.

One suitable approach would be to start with reading a chapter of the subject guide, followed by some of the further reading, if any of the material has not been understood or more information is desired. Then attempt the sample examination questions at the end of the chapter, which can be checked with the model answers and hints in Appendix B, before reading the learning outcomes. If you feel that these have not been fully achieved, go back to the further reading or to the web links, in order to make sure that you have fully understood each topic. You should also carry out the activities in each chapter, as they provide further insight to the topics being studied.

This subject guide makes use of a large number of acronyms. When a new acronym is introduced, it is expanded in full. Subsequent references will often just use the acronym. A list of acronyms used in the subject guide can be found in Appendix C.

1.5 Reading list

Because of the very varied material presented in this course, there is no one book that covers the whole course (or even the majority of it) and that can be recommended for essential reading. You may wish to obtain your own copy of one of the general networking books listed below or use library books and websites for further reading material.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, 2002.

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, 2003.

Kurose, James F. and Ross, Keith W. *Computer Networking.* (Addison-Wesley), third edition, 2005.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition, 2003.

1.6 Useful web links

The world wide web is a very dynamic medium and publishing a large number of web links in a subject guide is likely to result in the frustration of a 'File Not Found' response at some point in the future. Instead, an up-to-date list of useful links relevant to the course will be maintained on the CIS222 course web site at
http://doc.gold.ac.uk/~mas01pt/cis222/study/volume_2.htm.

In the event of this URL changing during the life of the subject guide, please follow the links to the author's home page from the staff page of the Goldsmiths' Department of Computing web site, and then follow the link to this page.

Two links are particularly useful:

http://www.rfc-editor.org/rfc.html – contains the full text of all the Internet standards which are known as Requests for Comments (RFCs).

http://www.protocols.com/protocols.htm – contains descriptions of most network protocols.

1.7 Study time

If you are studying this module full time then it should take approximately one quarter of your total study time for an academic year. In a typical institution you will probably spend about four hours per week on each module during term time in formal teaching. It is recommended that you spend at least three hours per week in reading and private study. You should also on average spend a further three hours per week in attempting sample examination questions at the end of each chapter and in doing the activities, plus formal coursework, in the lab or at home. If you are studying entirely on your own, then you should aim to spend an additional four hours per week in private study, to compensate for the hours that students in institutions receive formal teaching.

You should aim to spend a total of 300 hours on the whole module, including formal teaching. This total should also include time spent revising during reading weeks, vacations and prior to examinations. This times is applicable to an average student who aims to do well in the course. Some students who work more slowly may need to devote more time than this.

In order to complete this volume of the study guide in one (ten-week) term, you should aim to complete one chapter per week. This will leave two weeks spare for consolidation at the end or to allow for some slippage.

If you are studying part-time, over a longer period than one term, then you will have to adjust this recommended study time accordingly. Revision for examinations should be in addition to the above.

1.8 Examination

Important: the information and advice given in the following section are based on the examination structure used at the time this guide was written. However, the University can alter the format, style or requirements of an examination paper without notice. Because of this we strongly advise you to check the rubric/instructions on the paper you actually sit.

The examination will be a single three-hour written paper, usually sat in May, which will consist of a total of six questions. Three questions will be on the first half of the course (Data communications) and three on the second half of the course (Enterprise networking). You will be expected to answer four questions in all: two from the first half of the paper and two from the second half of the paper. Specimen examination questions can be found at the end of each chapter and a complete specimen examination paper can be found in Appendix A. Some model answers and hints can be found in Appendix B.

The overall mark for the course will be calculated from the examination results and the coursework marks.

Chapter 2: Basic concepts

Introduction

In this chapter we will take a brief look at the history of and define some basic terms used in data communications. We will examine Shannon's model of communications and classify different types of communications channels and study their characteristics. Finally, we shall study different types of networks and topologies.

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, (2003). Chapters 1.1–1.3, 3.5, 3.6.

Kurose, James F. and Keith W. Ross *Computer Networking.* (Addison-Wesley), third edition (2005). Chapters 1.2–1.3, 1.6.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition (2003). Chapter 1.1, 1.3, 3.4, 15.2, App 3A.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002). Chapters 1.1, 1.2, 2.1.3, 2.5.3.

2.1 Data communications

2.1.1 Brief history

Data communications is becoming an increasingly important part of everyday life, with the explosive growth of the Internet, third generation mobile networks that support multi-media applications and the increasing need for enterprises to communicate with their customers, their suppliers and their employees.

The requirement for data communications has resulted from a number of factors. Firstly, there is a need to exchange information. For most of mankind's history the speed at which information could be shared over a long distance depended on how fast a messenger could run or ride a horse. More sophisticated methods did evolve to transmit limited information such as the use of drums, smoke signals, beacons, semaphore signalling and carrier pigeons, but all of these systems had their limitations. The first recorded use of beacons is when Troy fell in about 1200 BC. Beacons are an example of extremely simple data communications that can signal just a single **binary digit** (**bit**) of information.

It was not until the mid-nineteenth century, with the widespread use of the telegraph and later the telephone and radio communications, that it became possible to communicate information between different parts of the world in real time.

It was some years after the development of the computer that the need to provide remote access to information arose. Initially, teletypewriter terminals were used to access mainframe computers to carry out remote job entry by inputting from paper tape readers and outputting to the teletype printer. Remote Job Entry soon evolved to using Card Readers and Line Printers over 300 and later 1,200 and 2,400 bit/s modem links.

In the 1970s minicomputers became popular, as they did not have to be located in large data centres, but could be sited in departmental offices under the control of user departments rather than centralised data processing departments. They often needed to be connected with similar machines in other departments or with the mainframe computers. Originally they emulated Remote Job Entry Terminals, but eventually the manufacturers developed proprietary architectures to connect these machines. The most common network architectures were the IBM[1] Systems Network Architecture (SNA) and the DEC[2] Digital Network Architecture (DNA).

[1] *International Business Machines Inc.*

[2] *Digital Equipment Corporation, now part of Hewlett Packard.*

When IBM launched the Personal Computer (PC) in 1981 it had no networking capability, but users soon discovered that they needed networks to transfer information between machines and to access corporate data held on mainframes. They started to connect their PC to mainframes using terminal emulation software and modems. Enterprises also discovered that, at the time, hard disk storage, printers and modems were very expensive if they were provided for each PC. They could be provided much more cheaply and efficiently as a shared resource. This led to the development of Local Area Networks (LANs) for fast access to high performance servers for file storage, sharing and print spooling and to modem pools. It also became more economic to run application software on these servers.

The next stage was for the LANs, minicomputers and mainframes to be linked up into an enterprise network, initially using proprietary architectures, but eventually moving to Internet standards.

Once PCs were properly networked, other services such as email began to develop which grew even more popular when enterprise networks began to link up to the Internet, and users were able to send emails to other organisations. The Internet connection also allowed access to the world wide web (www) and other Internet services.

Networks also can be used to improve the reliability of information services. Companies often have back-up computer facilities in a separate location to their operational centres. Networks can be configured so that traffic can be diverted to the back-up centre in the event of one of the operational centres failing. Networks also allow data to be easily replicated and backed up.

Networks have made distributed processing possible. Some grid applications such as the Search for Extra-Terrestrial Intelligence (SETI), where data is processed by thousands of PCs across the world, would not be possible if it were not for the ability to distribute processing via the Internet.

Networks also allow enterprises to monitor, manage and troubleshoot remotely located equipment including the network itself. Enterprises can save costs by centralising their technical staff.

2.1.2 Definitions

Data means distinct pieces of information[3] usually formatted in a special way. Data of itself has no meaning. It is only when it is processed by a data processing system that it takes on meaning and becomes information. Information is often coded as **digital data** expressed as a sequence of bits, but it can also exist as continuously varying **analogue data** such as in human speech and video. In this subject guide we will mainly be concerned with digital data.

[3] *Information was formally defined by Claude Shannon in his classical paper of 1948, but the mathematical theory of information is beyond the scope of this course.*

Data Communications is the transmission of data (usually assumed to be digital data) across distances. The distances can be very short, between a computer and a directly connected printer for instance, or can be very large,

on a national or global scale. When the distances are large the term telecommunications[4] is often used. Telecommunications often includes the transmission of analogue as well as digital data.

A **signal** is a means of transmitting an element of data across a distance and the process of doing this is called **signalling**. Signals can vary continuously (**analogue signals**) or have discrete levels (**digital signals**). As we will see in Chapter 8, there are four different combinations of analogue and digital data with analogue and digital signals that can be used.

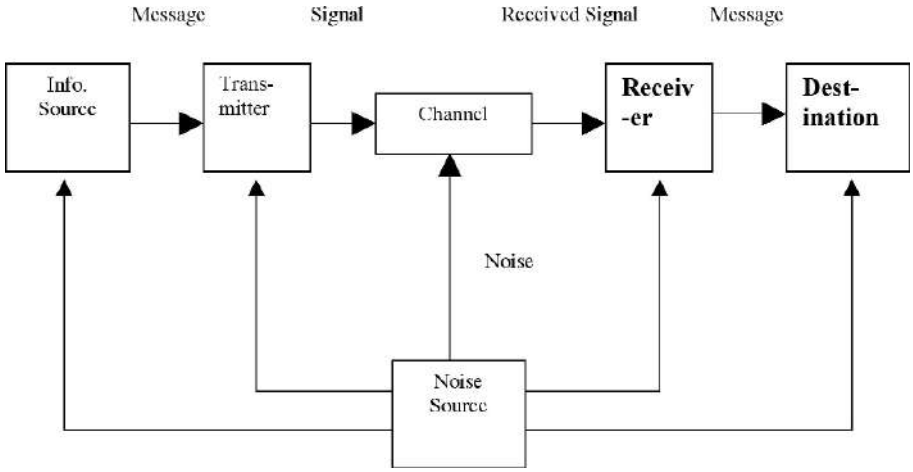
[4] *From the Greek word 'tele' meaning from far away or at a distance.*

2.2 Shannon's Communication Model

It is useful to start the study of the subject by examining a simple model of communications. Such a model was described by Claude Shannon in his classic 1948 paper.[5] Successful communication requires the following elements: an **information source** that produces information in the form of a **message**; a **transmitter** that converts the information into signals which are suitable for transmission through a **medium** (or **channel**) and a **receiver** that receives the signal and converts it back into the format of the original message which can be understood by the **destination**.

[5] *Shannon, Claude.* A Mathematical Theory of Communication. *(Bell System Technical Journal, 1948).*

Figure 2.1 Shannon's Communications Model



As can be seen from the above diagram, there is a complicating factor that affects the transmission of signals through the channel and systems. This is **noise**[6] which is generated by a **noise source**. All channels and systems are prone to experiencing noise to a greater or lesser extent. There is no such thing as a perfect channel or system. If a signal is amplified, then the noise will also be amplified. Engineers spend a great deal of time and energy in trying to design systems to minimise and overcome the effects of noise. All of the components of the model are affected by different types of noise but the greatest effect is upon the channel itself.

[6] *Noise is caused by spurious unwanted random energy generated inside or outside of the channel.*

2.3 Channels

Communications channels can be classified into a number of different types.

2.3.1 Quantitative characteristics

Channels can also be characterised by a number of different quantities:

- **Bandwidth:** the range of frequencies that can be effectively carried by the channel [measured in Hertz (Hz)]. The Hertz is a measure of the number of times a signal oscillates per second.
- **Capacity:** the maximum number of bits that can be carried by the channel in one second, often also called the **data rate** or **speed** [measured in bit/s].
- **Throughput:** the practical maximum number of bits that can be carried by the channel in one second as experienced by an end user [also measured in bit/s].

Table 2.1: Types of communication channels

Channel	Guided / Unguided	Analogue / Digital	Synchronous / Asynchronous	Simplex / Half / Full Duplex	Point-to-Point / Point-to-Multipoint / Broadcast	Switched Circuit Switched / Message Switched / Packet Switched	Multiplexed FDM/TDM/STDM	
Telephone call								
Mobile phone call								
Facsimile call								
TV mast to TV aerial								
TV remote control to TV set								
Dial-up modem to ISP								
Email between servers								
Web browser to server								
Private circuit								
Hub-based Ethernet								

- **Utilisation:** the proportion of time that the channel is fully occupied [measured in %].

Channels also exhibit some negative characteristics, which can be quantified:

- **Noise:** The causes of noise have already been discussed. Noise can be measured by comparing its power with that of the signal using a unit called the deciBel (dB). The deciBel is not an absolute unit. It is used to compare the power of two signals using a logarithmic scale.

  The **Signal to Noise Ratio** (SNR) is calculated using the formula:

  $SNR = 10 \log_{10} (S/N)$, where S is the power of the signal and N is the power of the noise (usually measured in Watts or milliWatts).

  The following table illustrates the logarithmic relationship between the actual ratio of signal to noise and the measurement of the Signal to Noise Ratio in deciBels.

- **Attenuation:** the loss of power of a signal over distance [measured in deciBels (dB)]

Table 2.2: Signal to noise ratios in deciBels

Signal : Noise	Signal to Noise Ratio (dB)
10,000 : 1	40
1,000 : 1	30
100 : 1	20
10 : 1	10
1 : 1	0

Here a comparison of two signal powers is made. The power of the transmitted signal is compared with the power of the received signal. A logarithmic unit (the deciBel) was chosen by engineers to make the computation of cumulative losses easier. The overall loss along a circuit can be calculated by simply adding all the individual losses on the components that make up the circuit. Without a logarithmic scale there would be a lot of multiplications and divisions. The deciBel was chosen rather than the Bel[7], as it allows virtually all losses and gains to be expressed as numbers between 0 and 100. The deciBel can also be used to calculate gains in power from amplifiers.

[7] *Named after Alexander Graham Bell, the inventor of the telephone.*

- **Delay:** the difference between the time a signal enters the channel at the transmitter and the time it exits the channel at the receiver [measured in milliseconds (ms)].

  There are different components that can contribute to delay. The ones we will be most interested in this course are:

- **Propagation delay**, which is caused by the limit imposed by the speed of light on any signal being transmitted in any channel. Nothing can travel faster than light (3 x 108 m/s) and typically communications signals travel in wires or optical fibres travel at about two-thirds of the speed of light (2 x 108 m/s).
- **Transmission delay**, which affects serial channels and results from the time it takes to actually transmit data bits serially onto a digital channel of a given capacity. If L bits are transmitted down a channel operating at a speed of R bits/s, then the transmission delay will be L/R seconds.
- **Queueing delay**, which can affect any time division multiplexed channel where data has to wait its turn to be transmitted. It is particularly important with packet switched channels.
- **Jitter:** the variability of delay [measured in standard deviations]. Minimisation of jitter is important for many real-time applications such as voice or video, as the presence of jitter will cause the received signal to break up, as continuity will be lost and ultimately the signal could become unintelligible.

Activity 2.1

In a spreadsheet, build a table to convert a Signal to Noise Ratio measured as a straightforward ratio to a Signal to Noise Ratio measured in deciBels. You should start with a ratio of 100:1, then 200:1 etc., up to 10,000:1.

2.3.2 Types of channels

Guided or unguided

In a **guided** channel, signals are constrained within the medium to follow a clear path. With **unguided** channels there are no (or few) constraints on how the signal propagates through the medium.[8]

[8] Metallic or fibre optic cables are examples of guided channels, while radio communications is an example of unguided channels.

Analogue or digital

Analogue channels carry analogue signals while **digital** channels carry digital signals. A digital channel is therefore ideal for carrying digital information where each bit, or sometimes a number of bits of data, can be represented by one of a number of discrete energy values used for transmission. In this subject guide we will mainly concentrate on digital channels, as they offer superior performance and, as a result, modern communications are almost exclusively digital.

Serial or parallel

In **serial** digital channels, data is transmitted a bit at a time, in sequence. In **parallel** digital channels, a number of bits are transmitted simultaneously over different physical paths. This subject guide will concentrate on serial channels, as parallel channels are usually only used for short distances such as on an internal computer bus or a printer cable.

Synchronous or asynchronous

In a **synchronous** channel, data arrives at fixed times and the receiver must always be kept in step (or be synchronised) with the transmitter. The receiver must know both the arrival time and the duration of each signal that is used to represent the bits in a large block of data. It must be able to run a clock that runs at the same speed as the clock in the transmitter. The two clocks are often synchronised by transmitting a clock signal, sometimes done by coding the data signal in a special way. In **asynchronous** channels, data can arrive at any time and the receiver does not always have to be kept in synchronisation with the transmitter. In this subject guide we will mainly concentrate on synchronous channels, as they are necessary for high speed communications and most channels operate synchronously today.

Simplex, half duplex or full duplex

A **simplex** channel only ever operates in one direction.[9] A **half duplex** channel operates in two directions, but only one direction at a time.[10] A **full** duplex channel can operate in both directions simultaneously.[11] Being able to do this is important for most applications and most modern communications is full duplex.

[9] An example of a simplex channel would be a channel used by a TV broadcast or a stock price feed.

[10] An example of a half duplex channel would be the channel used by a walkie-talkie radio.

[11] An example of a full duplex channel would be the channel used by a telephone call.

Point-to-point, point-to-multipoint or broadcast

A **point-to-point** channel has one transmitter and one receiver. A **point-to-multipoint** channel has one transmitter and a specified number of receivers.[12] A **broadcast** channel can have one or more transmitters and any number (often unknown) of receivers.

[12] Point-to-multipoint channels were used in old-style mainframe computer networks where the mainframe would poll a number of devices in turn on a single transmission line.

Baseband or broadband

A **baseband** channel is one where the channel is filled by just one signal[13] while a **broadband** channel can carry multiple signals at the same time, usually using Frequency Division Multiplexing.[14] Note that the term broadband can also be used to describe a high capacity channel (e.g. ADSL[15]) as opposed to a narrowband channel (e.g. a telephone channel).

[13] Ethernet is an example of a baseband channel.

[14] A Cable TV coaxial cable is an example of a broadband channel. Multiple TV channels are multiplexed together using FDM.

[15] Asymmetric Digital Subscriber Line.

Dedicated or shared

A **dedicated** channel is one in which only one end-to-end communication between different users is supported. A **shared** channel can support more than one end-to-end communication. Clearly, point-to-multipoint and broadcast channels are shared, but a point-to-point channel could either carry one end-to-end communication or multiple end-to-end communications. Dedicated channels are often more expensive and less efficient than shared channels as they make less efficient use of the channel.

Shared channels can result from:

- multiplexing or switching

  In order to carry more than one end-to-end communication, a channel can allocate its capacity to a number of different end-to-end communications simultaneously. This is known as a **multiplexing** channel.[16] Alternatively, a channel can allocate all its capacity to one end-to-end communication at a time, and when the communication has finished, allocate the channel to another end-to-end communication. This is known as a **switching**.

  Multiplexing can be achieved by Frequency Division Multiplexing (FDM), Time Division Multiplexing (TDM) or Statistical Time Division Multiplexing (STDM)

  - With FDM, capacity on the channel is shared by allocating a different range of frequencies to each end-to-end communication.[17]
  - With TDM, capacity on the channel is shared by interleaving data in individual timeslots for each end-to-end communication serially on a round-robin basis, but each end-to-end communication is allocated its own permanent capacity even when there is no data to be transmitted.
  - STDM is similar to TDM, but there is no fixed allocation of capacity to each end-to-end communication. They are allocated timeslots only when needed and can, if necessary, use extra spare timeslots. STDM is sometimes referred to as Asynchronous TDM (ATDM), as data on the input channels can arrive at any time.

Switching can be achieved by circuit switching, message switching or packet switching.

With **circuit switching**, an end-to-end communication seizes all the capacity of the channel for a fixed period of time by setting up a connection through the channel. With circuit switching, the channel can be idle while no data is being transmitted and the channel cannot be used by other end-to-end communications until the connection is released.[18] Circuit switching therefore does not usually make the most efficient use of the channel.

With **message switching**, an end-to-end communication consists of a single arbitrarily long message, which is transmitted through the channel, but as soon as the message has been transmitted, the channel is free to transmit another message from another end-to-end communication.[19]

Packet switching is similar to message switching except that packets cannot be of arbitrary length. Instead, messages are broken up into packets which all have a maximum size. Packets from different end-to-end communications can be interleaved on the channel.[20]

With message and packet switching the capacity of the channel is allocated much more dynamically than in circuit switching, and the channel is only idle when there are no end-to-end communications transmitting. The whole capacity of the channel can be allocated to just one end-to-end

[16] *In practice a channel can be both multiplexed and switched. It can be multiplexed into a number of sub-channels each of which could then be switched. An example of this is the 2 Mbit/s circuit used by network operators and enterprises to carry voice traffic. Each 2Mbit/s circuit is permanently divided into thirty 64 kbit/s voice channels onto which telephone calls are switched.*

[17] *For high-speed optical fibres, the term used for FDM is Wavelength Division Multiplexing (WDM).*

[18] *A good example of circuit switching is a 64kbit/s path between two telephone exchanges that can be allocated to any telephone call.*

[19] *E-mail as an application is an example of message-switching between mail servers.*

[20] *The Internet uses packet switching.*

communication, if it is the only one that requires capacity. Hence they are much more efficient, but delays also become variable and in the case of message switching can be very long.

Packet switching can be further subdivided into two distinct types of switching:

- **Virtual circuit** based (or connection-oriented switching) where a virtual circuit is set up through a network and all packets follow the same path through the network.[21] The effect of this is in some ways similar to circuit switching, but a virtual circuit uses no pre-allocated network capacity and hence is more efficient than circuit switching.

  Virtual circuits can be **switched** or **permanent**. Switched virtual circuits are set up as and when needed by means of a special call request packet which is allocated a virtual circuit number. This virtual circuit number is used in subsequent packets (instead of an address) to ensure that they follow the same route. When data transfer is complete the virtual circuit is disconnected and the virtual circuit number can be reused by other users. Permanent virtual circuits also use virtual circuit numbers but they are permanently set up by network administrators.

- **Datagram** based (or connectionless switching) where each packet is routed independently based on an address that the packet contains and each packet can follow a different path through the network.[22]

A channel can also be **composite**, built from a number of different components and media in series to provide a communications path between the transmitter and the receiver.[23]

[21] *Asynchronous Transfer Mode (ATM) is an example of a virtual circuit switched network.*

[22] *The Internet is an example of a datagram network.*

[23] *A good example of a composite channel would be a telephone channel, which will typically pass through many switches, multiplexers and copper and fibre optic communications circuits but will be able to carry a voice signal from the transmitting phone to the receiving phone.*

Activity 2.2

Consider different types of communication channels used in the various types of communication in Table 2.1 and any others you can think of. Attempt to classify them by type in the table. The last two columns only relate to channels that are shared in some way.

2.4 Networks

2.4.1 Definitions

A **network** is a collection of systems that are interconnected to exchange information with one another and share resources.

A **computer network** is a network that comprises terminals, computers, servers and other components (usually owned and managed together).

A **host** is a computer that runs (or hosts) end-user applications.

An **internetwork** or **internet**[24] is a collection of interconnected computer networks. Sometimes these networks are incompatible (running different protocols[25] and addressing schemes) and must be interconnected by Gateways, which translate between the different protocols. A **subnetwork**[26] (or **subnet**) is the part of a computer network left after all the hosts have been excluded.

- A **Wide Area Network** (**WAN**) is a computer network that covers a country or a continent or the whole world.
- A **Metropolitan Area Network** (**MAN**) is a computer network that covers a campus or a city.
- A **Local Area Network** (**LAN**) is a computer network that covers a limited geographical area (usually a single building or a part of a building).

[24] *The word internetwork or internet is a generic term and the global Internet (with a capital I) is a specific example of an internetwork.*

[25] *Protocols are sets of rules governing the procedures and data formats used for communication.*

[26] *Subnetwork also has a more specialised meaning in the Internet. Here a subnetwork (or subnet) can mean an individual network (such as an Ethernet) with a separate addressing range that forms part of an internetwork (such as the Internet).*

- A **Personal Area Network** (**PAN**) is a new type of computer network that is used to connect devices such as mobile phones, personal computers, personal digital assistants (PDAs) and peripherals, close to one person.

Large networks are also sometimes designed in a hierarchical structure with three clearly identifiable layers:

An **access network** is a network that supports end users. Access networks can be residential (e.g. local telephone, cable TV and ISP networks), institutional (e.g. a University or an office) or mobile where users are free to roam (e.g. a wireless LAN). Access networks are not normally resilient.[27]

[27] Resilient means being able to withstand failures.

A **distribution network** does not support end users but concentrates traffic from access networks and forwards it to a core network. Distribution networks often provide partial resilience.

A **core network** is a network that can switch or route large volumes of traffic. A failure of a core network will affect all users and hence it has to be highly resilient and stable.

2.4.2 Network topologies

Topology is a branch of Mathematics which is concerned with how things are connected. It takes no account of distances or spatial positions. It has been described as 'rubber band geometry'. Topology is highly relevant to the study of networks. Networks can be classified according to their topologies. There are a number of basic network topologies:

- **Bus networks** (Figure 2.2) are often used in LANs, but not MANs or WANs. All the nodes share access to a common medium. Buses are easy to implement but are not resilient to failures.[28] A cable fault will split the network in two. Isolating a fault on a bus is also very difficult but adding a new node is easy.
- **Ring networks** (Figure 2.3) are used for LANs, MANs and WANs.[29] They can provide resilience to single failures, if data can travel around the ring in either direction. Ring networks are harder to implement than bus networks and use more cable, but faults are much easier to isolate. Adding a new node is difficult.
- **Star networks** (Figure 2.4) are used in both LANs and WANs.[30] They are not particularly resilient to failures, especially at the hub or central site. It is relatively easy to add new nodes, and star networks are easy to fault and manage as most of the network equipment is centralised.
- **Tree networks** (Figure 2.5) are really extended buses; they could actually be described as Bus-Bus[31] hybrid networks where there is a single medium which branches in various places (not at nodes).[32] They possess little resilience and fault isolation is difficult, but new nodes can easily be added.
- **Mesh networks** are normally used in WANs and they are relatively easy to manage and fault. They come in two varieties:
  - **Fully-meshed networks** (Figure 2.6) are usually only economic for small networks or in core networks. They are extremely resilient to multiple failures. They are very expensive if there are a large number of nodes. Adding a new node is difficult and expensive.
  - **Partially-meshed networks** (Figure 2.7) are resilient to multiple failures depending on the degree of connectivity. They are not as expensive as fully meshed networks. Adding a new node is somewhat easier and less expensive than for a fully-meshed network.

[28] The original thick and thin Ethernets use bus topologies. All the stations were connected into a coaxial cable which was run around a building.

[29] IBM Token Ring LANs and modern wide area transmission systems such as Synchronous Digital Hierarchy (SDH) use ring topologies.

[30] Twisted pair Ethernets and mainframe computer networks, where remote terminals access a central site, use star topologies.

[31] See terminology used to describe hybrid networks.

[32] Bridged Ethernet LANs and legacy mainframe computer networks use tree topologies; the latter via multidrop (point to multipoint) circuits.

- **Hybrid networks** are composed of two or more of the basic types and inherit the advantages and disadvantages of their component topologies. They are often described by hyphenating the basic types, with the second term indicating the topology at the centre of the network:
  - **Star-Bus networks** (Figure 2.8) have a central bus which connects together a number of star networks.[33]
  - **Star-Star networks** (Figure 2.9) are commonly known as Cascaded Star Networks as there is one central star network to which other star networks are connected.[34] This topology has little resilience, particularly if all traffic has to pass through the central node.
  - **Star-Ring networks** (Figure 2.10) have a central ring which connects a number of star networks.[35] It has a resilient core but does not provide resilience to each outlying node.
  - **Star-Mesh networks** (Figure 2.11) have a central mesh network to which a number of star networks are connected.[36] It has a very resilient core but does not provide resilience to each outlying node.
  - **Ring-Ring networks** (Figure 2.12) are also known as Multiple Overlapping Rings (or Shared Protection Rings or SPRings). The topology is a partial mesh network, but is constructed in a special way. It comprises a set of rings with shared segments.[37] It provides a high degree of resilience to multiple failures at low cost.

[33] Star-Bus networks are sometimes used in LANs where each floor has a star-connected Ethernet and the hub on each floor is connected to a Fast Ethernet bus, which connects all the floors.

[34] Star-Star networks are used in old-style mainframe WAN networks where cluster controllers were positioned in such a way as to support remote terminals with minimum circuit costs.

[35] Star-Ring networks are often used in LANs where each floor has a star-connected Ethernet and the hub on each floor is connected to a ring, which connects all the floors.

[36] Star-Mesh networks are used in large WANs where there is a partially (or fully) meshed core connected to distribution networks in a star configuration.

[37] SPRings have become increasingly popular with the telecommunications operators who use them for their Synchronous Digital Hierarchy transmission networks.

Figure 2.2: Bus Topology

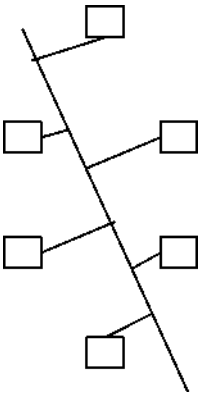


Figure 2.3: Ring Topology

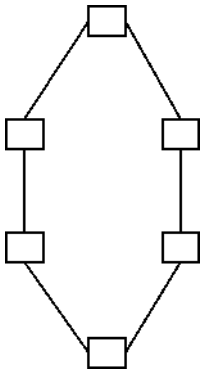


Figure 2.4 Star Topology

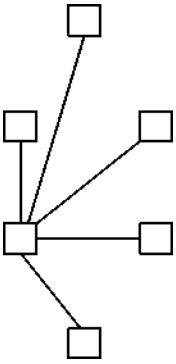


Figure 2.5 Tree Topology

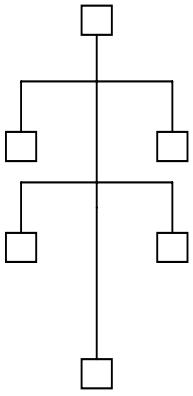


Figure 2.6 Full Mesh Topology

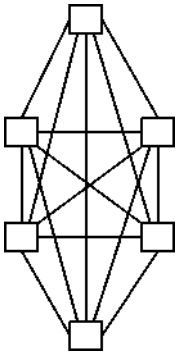


Figure 2.7 Partial Mesh Topology

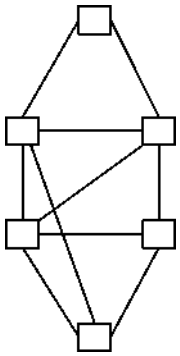


Figure 2.8: Star-Bus Topology

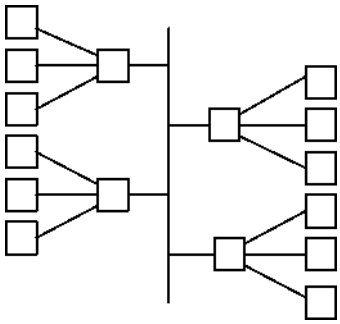


Figure 2.9 Star-Star Topology

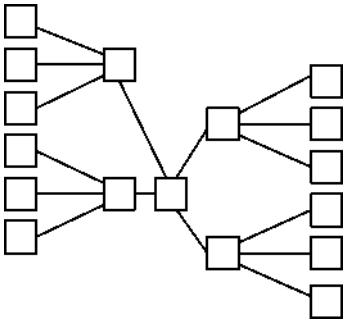


Figure 2.10: Star-Ring Topology

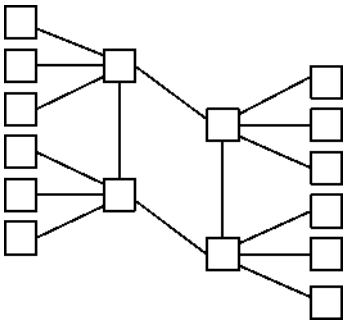


Figure 2.11: Star-Mesh Topology

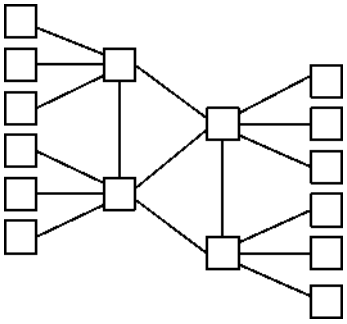
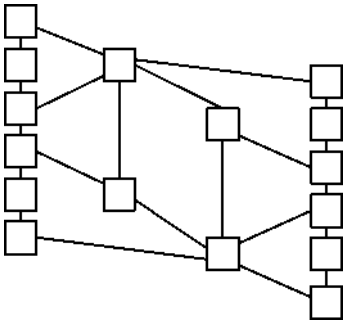


Figure 2.12: Ring-Ring networks



Activity 2.3

Investigate the topologies of the local area and wide area networks that are deployed by your educational institution, business or other enterprise in your locality. You may need to talk to a technician or ask to see some network maps. Which network topologies are employed and why? How resilient are the topologies to failure?

Activity 2.4

Explore the Atlas of Cyberspace web site (http://www.cybergeography.org/atlas) and examine and attempt to classify the topologies used by Internet Service Providers.

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
    i. Noise affects all the components of Shannon's Communications Model and not just the channel.
    ii. All packets using virtual circuits contain a full network address.
    iii. Propagation delay results from the time it takes to actually transmit data bits serially onto a digital channel.
    iv. A sub-network is the part of computer network left after all the hosts have been excluded.
b. Describe the main differences between synchronous and asynchronous channels.
c. Explain why jitter is undesirable for real-time communications such as voice and video communications.
d. A channel experiences an average noise power of 0.2 mW and the average power of the signals it carries is 200 mW. What is the Signal to Noise Ratio in deciBels? If the signal is amplified by 40 dB, what will be the average power of the noise?
e. For a WAN consisting of 6 nodes, draw the topology with the least number of point-to-point links and draw a topology that best improves the resilience of this network by adding a single link. Draw another topology that will maximise resilience for the whole network and another that will minimise delays between one node and all of the others.

Learning outcomes

At the end of this chapter and the relevant readings, you should be able to:

- outline the history of data communications
- describe Shannon's Communication Model and all its components, including the various sources of noise which affect all communications channels
- explain the qualitative and quantitative characteristics of a channel
- classify communication channels according to their characteristic type
- calculate gains, losses and Signal to Noise Ratios in deciBels
- classify networks according to their type
- identify different basic and hybrid network topologies and list their advantages and disadvantages.

Chapter 3: Network architecture

Introduction

In this chapter, we will examine what is meant by layered network architectures and why they are needed. We will study the different types of standards and the organisations that produce them. We shall consider how network devices can be classified in relation to the hybrid reference model and to their role within the network. Finally, we shall examine the concept of a protocol and introduce some general terminology that can be used in describing the operation of protocols.

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, (2003). Chapters 1.4, 2.1, 2.2, App 3.

Kurose, James F. and Keith W. Ross *Computer Networking.* (Addison-Wesley), third edition, (2005). Chapters 1.7.1.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition, (2003). Chapters 0.3, 2.1.2, 2.3, 2.4, 18.1.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002). Chapters 1.3, 1.4, 1.6, 4.7.5.

3.1 Layered architectures

When building complex structures or systems, engineers like to define an architecture. Computer hardware, software and networks are easier to design, build, test and maintain if they are based on a well defined framework or architecture. A **network architecture** can be defined as a highly structured framework within which networks can be analysed, designed and implemented, incorporating a defined set of layers and protocols.

Computer and network architectures are defined as a set of layers which perform different functions and can be implemented separately with well defined interfaces between each layer. The lower layer provides a **service** to the layer immediately above it by means of an **interface**. Think of a layer as software implementations that perform specific functions and the interface as a set of procedure calls with parameters. With network architectures there must be two similar implementations at both ends of a communications channel. These layers are called **peers** and they communicate with each other using a **protocol** (or set of rules) specific to that layer. The set of layers is often referred to as a **protocol stack** or **protocol suite**.

With layered architectures:

- each layer is functionally independent
- each layer has a well defined interface to the previous layer
- each layer communicates indirectly with its peer layer at the opposite end of a communications channel using a protocol specific to that layer
- each layer communicates with its peer layer via an interface with the layer immediately below, unless it is the bottom layer, in which case it communicates directly with its peer.

The advantages of layered architectures are as follows:

- they divide complex operations into more manageable groups which are then more easily implemented and tested
- it is possible to change one layer without affecting all the others, providing that the same interfaces are supported[1]
- it is possible to mix and match different technologies and suppliers for different layers if they support standard interfaces.[2]

The advantages of layered architectures are clear; the only question is how many layers are needed and what functions are to be placed in each layer. There are two main models for defining network architecture. The International Organization for Standardization (ISO) has defined a seven layer model called the Open Systems Interconnection (OSI) Reference Model. This model was developed in the 1980s as part of an exercise to produce open standards to free users from the proprietary standards which were being used by computer vendors to lock customers into their network architectures. At the time the vast majority of customers were using IBM's proprietary architecture called Systems Network Architecture (SNA) which also had seven layers. OSI was not particularly successful. Both OSI and SNA were soon overtaken by the development of the Internet. The researchers working for the US Military who developed the Internet used their own four layer model called the Internet or Department of Defense (DoD) Model. This model is also sometimes called the TCP/IP Reference Model, named after the two main Internet protocols.[3] Tanenbaum gives a good comparison of the two models and an interesting insight into the reasons why the Internet protocols now predominate.[4]

Both ends of a communications channel must implement the same standards for each layer in order to communicate. These standards are called **protocol standards**. They define the syntax of the data to be transmitted at this layer and all of the other rules which govern communications between peer entities at this layer. Communication between the layers at both ends of the channel is not direct (apart from at the very bottom layer). Instead each layer relies on the layers below it to effect communications. It makes use of the services provided by the layer immediately below which may in turn rely on services provided by another layer below. In effect, what actually happens is that each layer adds a **protocol header** to the data that is to be sent. A protocol header contains information that is only relevant to the peer layer at the other end of the channel. When a packet of data is received by the peer layer at the other end of the channel, the protocol header is processed and then stripped away before the data is passed to the layer above. In this way the upper layers are not aware of or do not need to know anything about the protocols or headers being used by lower layers.

Before we look at network architecture in detail, it will be useful to define some commonly used terms. An **interface** exists between two adjacent layers through which the lower layer provides a service to the upper layer. The interface is formally specified by a set of **primitives** which are implemented as procedure calls with parameters and which are executed whenever the upper layer requests a service from the lower layer.[5] The **Service Access Point (SAP)** is the place at which the service is provided and is identified by a unique **Service Access Point (SAP) Address**.[6]

[1] For example, a carrier would upgrade its network layer from IP version 4 to IP version 6 without having to make any changes in the transport layer above or the data link layer below.

[2] For example, a Local Area Network could be changed from an IBM Token Ring to an Ethernet at one site without affecting the Internet Protocol and application layer software that generates and receives data, or without requiring a change to the LAN used at any other site with which the first LAN communicates.

[3] Transmission Control protocol / Internet Protocol.

[4] Tanenbaum, Chapter 1.4.3, p. 44.

[5] E.g. The IP network service has just two primitives: SEND to send a packet throughout the network layer and DELIVER to receive a packet from the network layer.

[6] E.g. The Network Service Access Point for the Internet Protocol is identified by a unique Network Service Access Point (NSAP) address. This NSAP address for IP is more commonly known as the IP address.

3.2 Reference models

You studied the Open Systems Interconnection (OSI), Internet and Hybrid Reference Models in a Level 1 unit. You should re-familiarise yourself with the functions of the layers in each model. The names of the layers are summarised in Table 3.1, below.

Table 3.1: Comparison of OSI and Internet reference models

OSI	Internet	Hybrid
Application Presentation Session	Application	Application
Transport	Transport	Transport
Network	Internet	Network
Data Link Physical	Network Access	Data Link Physical

This course will assume the hybrid model which attempts to make use of the best features of both models to produce a practical and simple model that can be used to describe network architectures. This model has five layers and its relationship with the OSI and Internet models can be seen in Table 3.1.

Activity 3.1

Look at the web site http://www.protocols.com/protocols.htm. You will see a table with a long list of protocols in the first column. Find some protocols that you have heard about and click on the link to the family of protocols that these protocols belong to. If you have not heard of any of the protocols then click on some of the families of protocols in the third column of the table. For most families, by scrolling down the page, you will be able to see a table that shows how the protocols in that family relate to the OSI Reference Model.

3.3 Network standards

One word that has occurred a number of times so far in this section is 'standard'. Standards allow different vendor products to inter-work. They can be *de jure* (by law) if they have been produced or accepted by a recognised standards body or *de facto* (by fact) if they have not been produced or accepted by a recognised standards body, but have gained widespread use by market forces. Hardware and communications standards are often *de jure* while software standards are often *de facto*. A good example of a *de facto* standard is the Microsoft Windows operating system. Competitive markets will only usually arise after *de jure* standards have been agreed and have stabilised. Network vendors work together in standards bodies to agree standards for new technologies. *De facto* standards can be further divided into proprietary (or closed) standards which are produced, owned and controlled by a commercial organisation (e.g. Microsoft Windows) and open standards which, while they may have been originally produced by a commercial organisation, have since been transferred to the public domain (e.g. Unix).

There are a large number of standards bodies which develop standards for data communications and networks. The main international standards bodies are:

- International Organization for Standardization (ISO) whose members are various national standards bodies such as the American National Standards Institute (ANSI) in the USA, the British Standards Institute (BSI) in the UK, Association Française de Normalisation (AFNOR) in France, and Deutsches Institut für Normung (DIN) in Germany.

  ISO produces a wide range of standards in many areas other than telecommunications. Its standards are prefixed by ISO.[7]

- International Telecommunications Union Telecommunications Standardization Sector (ITU-T) whose members are national governments, regulatory agencies and various equipment suppliers and network operators. It is a part of the United Nations and its standards, called recommendations, are prefixed by a letter of the alphabet.[8]
- European Technical Standards Institute (ETSI) whose members are administrators, manufacturers, network operators, service providers and users, in Europe and other countries. It produced the GSM standard for second generation mobile phones. Its telecommunications standards are prefixed by ETSI EN.[9]

In addition to these international standards bodies, there are a number of other bodies which set standards in specific areas of networking. These include professional bodies and forums from industry and academia interested in developing new standards. Examples of these bodies are:

- Institute of Electrical and Electronic Engineers (IEEE) which has been particularly active in the standardisation of LANs. Its standards are prefixed by IEEE.[10]
- Internet Engineering Task Force (IETF), which is responsible for all standards relating to the Internet. Its standards are prefixed by RFC which stands for Request For Comment which continues to be used after a standard has been agreed.[11]
- Electronic Industries Association / Telecommunications Industry Association are two American trade associations which have been accredited by ANSI to develop standards for the electronic and telecommunications industries. They are both involved in developing computer/network interface standards. Their standards are prefixed by EIA/TIA.[12]
- Frame Relay Forum which is an industry group which was set up to standardise Frame Relay protocols. Standards, called implementation agreements, are referenced by numbers prefixed by FRF.[13]
- ATM Forum which is an industry group which was set up to standardise Asynchronous Transfer Mode protocols. ATM standards are referenced by numbers prefixed with af.[14]

[7] *For example, ISO 4335.*

[8] *For example, G.723, Q.931, V.24, X.25.*

[9] *For example, ETSI EN 301 419–1.*

[10] *For example, IEEE 802.3.*

[11] *For example, RFC 791.*

[12] *For example, EIA/TIA 232F.*

[13] *For example, FRF 1. 2.*

[14] *For example, af-phy-0015. 000.*

Activity 3.2

Use a search engine to find the websites of the standards bodies referred to in this chapter, and explore these sites to see what standards work they have been engaged in.

3.4 Network devices

Network devices can be classified according to the highest layer at which they operate. This layer is the highest layer for which they need to be able to read the protocol headers. All network devices will be able to forward application

layer data, but they do not have to understand what the application layer headers and data mean. A device is therefore regarded as a network layer device if it has software that enables it to read network layer headers, but not transport layer headers, in order to carry out its function. A network layer device will, of course, require software to handle the data link and physical layers.[15]

The hosts will always require a full protocol stack and will need software to read application headers and data as well as all the other headers in the protocol stack (See Fig 3.1). Other devices may only need to be able to read headers from network layer and below (See Fig 3.2), and others will only need to read data link headers (See Fig 3.3). Some devices will not even need to be able to read the data link headers. These will be physical layer devices (See Fig 3.4).

[15] *As an example an IP Router is a network layer device, as it has software that enables it to read the IP address in an IP header in order to route an IP packet. But it also has software on its line cards to handle Ethernet and serial ports running different data link protocols. It may receive a frame from an Ethernet, read the data link address to check that it is the correct destination for the frame, then read the IP header to find the destination of the packet and look this up in its routing table to discover which port should route the packet. Finally, the line card which supports this port has to encapsulate the IP packet in the data link protocol appropriate to that port. This process is shown conceptually in Figure 3.2.*

Activity 3.3

Explore the RFC Search web site (http://www.rfc-editor.org/rfcsearch.html) and perform an RFC search for some protocols that are used on the Internet. Examine a few of the RFCs that specify these protocols to see how they are written.

Figure 3.1: An application layer intermediate device between two hosts

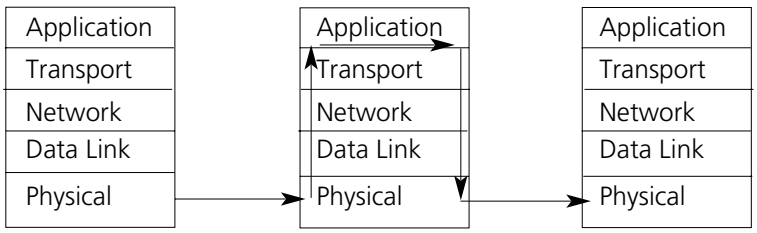


Figure 3.2: A network layer intermediate device between two hosts

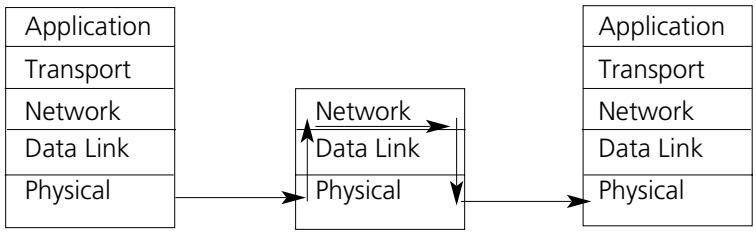


Figure 3.3: A data link layer intermediate device between two hosts

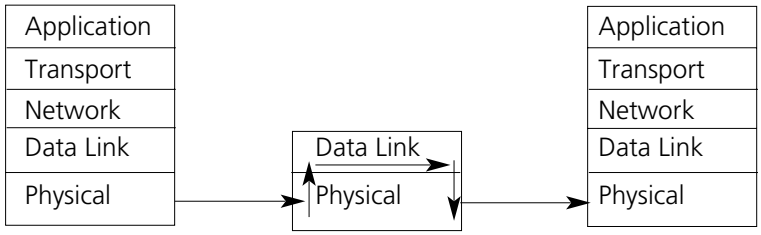
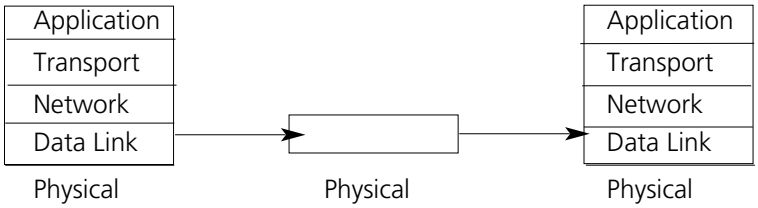


Figure 3.4: A physical layer intermediate device between two hosts



It should be noted that because transport protocols operate end-to-end there are very few requirements for intermediate transport layer devices. There are however many intermediate application layer devices in common use.[16]

Another classification of network devices is whether they are **DTE**s or **DCE**s. DTE stands for **Data Terminal Equipment** and DCE can either stand for **Data Communications Equipment** (according to the EIA) or **Data Circuit-terminating Equipment** (according to the ITU). The distinction originates from the need to have different interface specifications at either end of the cable between a data terminal (such as a PC) and a modem.[17] The PC is configured as a DTE and the modem as a DCE. Another important difference between the two occurs when the modem supports synchronous communications; the modem must provide a timing signal to the PC to maintain synchronisation. Another key distinction is that the interface on a DTE is normally male and the cable connector that plugs into it must be female. DCEs normally have a female interface and the cable connector that plugs into them must be male.

As a general rule, equipment that historically belonged to or still belongs to network operators is usually configured as a DCE and equipment belonging to their customers (that is a source or destination of data) is configured as a DTE. Most modems are now owned by customers, but they must still be configured as DCEs. You may think that this distinction is not very important, but when you try to connect two different local devices with a cable, it is crucial. If one device is a DCE and one is a DTE, then they can be connected by a straight-through male-to-female cable where each pin on one connector is connected via the cable to an equivalent pin on the other connector. However, if both devices are DTEs[18] or both devices are DCEs, this straight-through cable will not work and a cross-over cable is required, where some pins on one connector are not connected to the equivalent pins of the other connector. On many network devices, such as routers, the ports can be configured by software either to be DTEs or DCEs.

[16] *A good example of an intermediate application layer device, is a mail relay that reads e-mail headers and decides where the message should be forwarded.*

[17] *On a standard EIA232 interface pin 2 is called transmit and pin 3 is called receive, but only one end of the interface cable can transmit on a pin 2. The other end must receive on pin 2. Likewise with pin 3. If both ends were configured in the same way, communication would be impossible. In this case, the device that transmits on pin 2 is the DTE and the device that transmits on pin 3 is called the DCE.*

[18] *If you try to connect up two PCs back to back using a cable, a female-to-female cross-over cable is required as both will be configured as DTEs.*

Activity 3.4

Using textbook indexes or web searches, read about the network devices in Table 3.2 and complete the table by determining the layer of the hybrid model at which they operate and whether they are normally configured as DTEs or DCEs.

Table 3.2: Network devices

Device	Layer	DTE/DCE
Amplifier		
Asynchronous Transfer Mode (ATM) Switch		
Bridge		
Digital Subscriber Line Access Multiplexer (DSLAM)		
Frame Relay Access Device (FRAD)		
Front End Processors (FEPs)		
Gateway		
Hub		
Inverse Multiplexer		
Local Area Network (LAN) Switch		
Modulator/Demodulator (Modem)		
Multiplexer		
Network Terminating Equipment (NTE)		
Repeater		
Router		
Packet Assemblers/Disassembler (PAD)		
Packet Switch		
Private Automatic Branch Exchange (PABX)		
Telephone Switch		

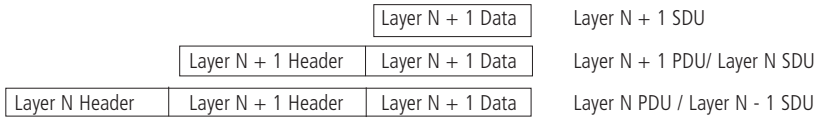
3.5 Network protocols

There are also specialised names used to describe data packets at various points in the layered network architecture. When layer N+1 wishes to communicate with its peer layer, it does so by transferring a packet known as a layer N+1 **Service Data Unit (SDU)** across the interface with the layer N immediately below, along with some control information. Layer N then adds some protocol header to the Layer N+1 SDU to form a larger packet called a layer N **Protocol Data Unit (PDU**), which consists of the layer N **header** consisting of a number of different fields used by the protocol, followed by the layer N **data field**[19] which is actually the layer N+1 SDU. The layer N header contains information only relevant to layer N and it is sent just ahead of the

[19] *The data field is also often referred to as the payload.*

data. The PDU is transferred to the peer layer N entity, possibly via lower layer SDUs and PDUs. The peer layer N entity then strips off and processes the layer N protocol header and passes the SDU up to layer N+1.

Figure: 3.5: SDUs and PDUs



Instead of using numbers to describe layers, names and letters are often used. For instance, in OSI terminology, a Network Service Data Unit (NSDU) will be passed from the Transport Layer to the Network Layer via a Network Service Access Point (NSAP) which will add appropriate Network Layer headers to it to create a Network Protocol Data Unit (NPDU). This will be forwarded to the peer Network Layer, which will then strip off the Network Layer headers and pass the NSDU up to its Transport Layer.[20]

In this subject guide we will use the more common terminology and use the word **message** to describe an Application SDU/PDU; **segment** to describe a Transport SDU/PDU; **packet** to describe a Network SDU/PDU; and **frame** to describe a Data Link SDU/PDU.

[20] Or even more simply using TCP/IP terminology; a TCP segment is passed from the TCP layer to the IP layer via the IP interface, which adds an IP header to it and then transmits it as an IP datagram.

The interfaces between layers (Service Access Points in OSI terminology) are implemented as a set of **primitives**. These can be thought of as a set of software procedures or functions with parameters. One of these parameters will be the Service Data Unit to be transmitted, or more accurately a pointer to a memory buffer that contains the SDU. OSI primitives are of four basic types as follows. A **Request** is issued from a higher layer to a lower layer to request a service, which is then carried by a PDU to the peer layer whereupon it is passed up to the higher layer as an **Indication**. The higher layer at the distant end then issues a **Response** to the Request to its lower layer which is again transferred via a PDU to the requester where it is passed back up to the higher layer as a **Confirmation**.

Network architectures are defined by a set of protocol standards for each layer in the reference model. The standards are developed and agreed by various standard bodies.

A **protocol** is a set of mutually agreed rules that allows two peer layer entities to communicate with each other.

Protocols can be as follows.

3.5.1 Symmetric/asymmetric

Some protocols are completely symmetric. The implementation at both ends of the communications channel is identical and all the functions provided by the protocol can be invoked from either end. These protocols are often referred to as peer-to-peer. Other protocols are asymmetric where one end can invoke functions that the other end cannot, in which case, the protocol is often referred to as a master/slave or a client/server protocol.

3.5.2 Standard/proprietary

Protocols can either be standard or proprietary. Standard protocols are controlled by official standards bodies, whereas proprietary protocols are controlled by commercial organisations. Customers today are wary of proprietary protocols, as their use tends to lock them into suppliers.

3.5.3 Connection-oriented/connectionless

This is an important distinction. A connection-oriented protocol requires that a connection is set up between both ends before any data can be transmitted. With a connectionless protocol, data can be sent at any time without any need to set up a connection beforehand. With a connection-oriented protocol some state information regarding the connection will be required at all the nodes involved in the connection, and if this state information is lost due to a node crashing the connection will disappear and will have to be re-established. Connectionless protocols do not suffer from this problem and are hence more resilient. Because of this connection-oriented protocols are sometimes called stateful protocols and connectionless protocols are often called stateless protocols. Connection-oriented protocols do, however, have some advantages. They can be more efficient, as protocol headers can be shorter, because each data packet does not have to be individually addressed. Also because there is a connection set-up, it is possible to allocate resources to support the connection at every node and hence the quality of service can be guaranteed. In other words, all data packets can be guaranteed to arrive on time, in order and without loss or duplication. It is not possible to guarantee quality of service with connectionless protocols.

Protocols in different layers of a network architecture will have many different functions, but a large number of protocol functions appear in similar forms in many of the layers. They are based on the generic protocol functions listed by Stallings[21] and are listed below:

[21] *Stallings, pp. 575–580.*

3.5.4 Encapsulation/de-capsulation

This function has already been discussed, although the terminology is new. A layer receives an SDU from the layer immediately above, adds its own protocol header to the SDU, which is encapsulated in the PDU and then transmitted to the peer layer, which strips off the header and passes the de-capsulated SDU to the layer immediately above.

3.5.5 Segmentation/reassembly

It is sometimes the case that the SDU received from the layer immediately above is too big to be carried in a PDU. In this case the layer must segment the SDU into one or more PDUs and send these separately. The peer layer must recognise such segmented PDUs and reassemble them into the original size SDU before passing them up to the layer immediately above. In the case of IP, this process is called fragmentation rather than segmentation, as TCP uses the term segment to describe its PDUs. In TCP/IP, a TCP segment that is too large for an IP network layer to transmit in one packet is fragmented into one or more smaller packets and is reassembled by the peer network layer. Protocols which support segmentation will contain some kind of sequence number field in their headers to assist in reassembly.

3.5.6 Multiplexing/de-multiplexing

One layer can often carry many SDUs between several different pairs of communications entities in the layer above to make efficient use of lower level services. This is called multiplexing. The PDUs must contain information (usually addresses) that identifies the entities involved so that the peer layer can de-multiplex PDUs to the correct entity.

3.5.7 Addressing

In order to communicate with an entity at any layer, it is necessary that the entity can be uniquely identified. Most PDU headers contain an address field which uniquely identifies the peer entity to which it is communicating.3.5.8

3.5.8 Ordered delivery

With connectionless protocols, PDUs can arrive out of order as they may take different routes. When this happens the layer must buffer and reorder the PDUs higher.

3.5.9 Priority

It is sometimes necessary for certain PDUs to be given preferential treatment in queues so they can be transmitted with minimum delay. This may be determined by a higher layer, such as an application that operates in real time and thus has a requirement for low delays. It may also be used to allow certain higher level commands, such as those that will interrupt a process on the destination to overtake some of the data that would otherwise arrive beforehand and be processed. To indicate the priority of a PDU, there is often a field in the PDU header that will indicate its priority from a number of different levels.

3.5.10 Error control

PDUs can be corrupted or lost in transmission. Fields in PDU headers such as sequence numbers or error detection codes can be used to detect errors. Once errors are detected, some protocols will attempt to recover from the errors. Other protocols (especially connectionless ones) will simply discard PDUs in which errors are detected.

3.5.11 Flow control

Sometimes a transmitter can send PDUs at a faster rate than a receiver (or even the network itself) can process them. In such situations it is useful to have a flow control mechanism to regulate the speed of transmission. This is often achieved by means of a procedure whereby PDUs referenced by a sequence number in the protocol header are acknowledged. Connectionless protocols usually do not support flow control, but will simply discard PDUs if they are arriving at too fast a rate to be processed.

3.5.12 Connection control

For connection-oriented protocols, a mechanism must exist to establish and close a connection. This is usually achieved by means of special PDUs, whose meaning is defined in a type field in the protocol header, to request the establishment of a connection or to request an existing connection be closed. In order to ensure a consistent state at both ends of the connection, the requests will have to be acknowledged by another special PDU indicated by its type field.

3.5.13 Grade of service

When establishing connections, it is useful to specify what performance is required from the connection in terms of such parameters as delay, jitter or throughput. These parameters can define the grade of service required for the connection. The PDU that establishes a connection will often contain a field to specify the grade of service. If the receiver and the network can

support the grade of service requested, the PDU will be positively acknowledged. If the grade of service cannot be supported the required grade of service parameters can be negotiated downwards or the connection could be immediately closed.

3.5.14 Security

A number of protocols contain built-in security features to perform functions such as authenticating the parties involved in the communication. This is also achieved by means of authentication fields in the PDU headers.

3.5.15 Data encoding

Protocols must define the syntax (format) of data fields in PDUs as well as the semantics (meaning). The rules for how data in a PDU is represented have to include such things as the type of the data[22] and the length of the field in bits or bytes. The semantics of the data is defined in rules that ascribe meaning to each allowable value of the data field.

[22] *For example, integer, character string, etc.*

3.5.16 Data encryption

Data encryption is closely associated with security, but can relate to either the whole PDU or just the data field of the PDU, which needs to be scrambled so that parties other than the recipient (who alone has the facility to decrypt the data) cannot view the data being sent.

3.5.17 Data compression

Transmitting some types of data can be very inefficient. Transmitting bit mapped images, full video and audio all require large volumes of data to be transmitted and hence will require either large transmission capacity or a long transmission time. However, these types of data contain a large amount of redundant information that can be compressed (often without a perceptible loss of quality) so that it can be transmitted more efficiently. Some protocols provide a capability to compress data.

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
    i. The application layer in the hybrid model corresponds to the top two layers of the OSI Reference Model.
    ii. ITU-T standards are called recommendations.
    iii. When connecting a PC to a modem, the PC has a DTE interface and the modem a DCE interface.
    iv. Protocol Data Units are passed across the interface between adjacent layers in a host.

b. What is meant by a layered architecture and what advantage is obtained from using such an architecture?

c. List the full expanded names of three standards bodies and give an example of a standard that each body has defined.

d. Describe the main differences between connection-oriented and connectionless protocols.

e. Describe how protocols are encapsulated and decapsulated.

Learning outcomes

At the end of this chapter and the relevant reading, you should be able to:

- list the characteristics and advantages of layered architectures
- identify the main standards bodies involved in the development of protocols and which types of protocols they specify
- distinguish between de jure and de facto standards
- identify which network devices operate at which layer of the Hybrid Reference Model, and which are normally configured as DTEs and which as DCEs
- distinguish between the characteristics of a DTE and a DCE
- describe how data is transmitted within a layered architecture using correct terminologies
- distinguish between standard and proprietary protocols, symmetric and asymmetric protocols and connection-oriented and connectionless protocols
- describe the generic functions of protocols which can be applied to protocols in any layer.

Chapter 4: The application layer

Introduction

In this chapter, we shall examine the services and interfaces provided by the application layer and the general functions of application layer protocols. We shall then examine in some detail virtual terminal, web, email, file transfer and access, directory and network management protocols.

A very large number of application layer protocols have been designed for many different applications. We will only examine in detail a small number of protocols that are commonly used on the Internet, plus a few ISO protocols in outline.

This subject guide will not cover all the details of protocols (either the fields in the protocol headers or the protocol mechanisms). Rather it will attempt to summarise their basic functions. You are advised to study in detail the main protocols used on the Internet. Details of these protocols can be found in any networking textbook, or from looking up the RFCs, or from websites.

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, (2003). Part 6, Chapters 24.1, 25, 26, 27.

Kurose and Ross, *Computer Networking – A Top Down Approach featuring the Internet* (Addison Wesley), third edition (2005). Chapter 2.1.1, 2.2.1–6, 2.3, 2.4.

Stallings, William, *Data and Computer Communications.* (Prentice Hall International), seventh edition (2003). Chapter 22.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002). Chapters 5.4.1, 7.1–7.3.

4.1 Services

The application layer is the layer that provides communications functions for a network application to serve an end user or another application program.

If an application layer entity is providing a service direct to an end user, then the software that provides the interface between the end user and the networked application is described as a **user agent.**[1]

Network applications are often implemented as **client server systems**. Under the client server model, the client only runs when it is required and initiates a request to the server and the server replies with a response.[2] The server will typically handle requests from many clients and will run continuously. Both the user agent (client) and the server run **application processes** that work together via a network to deliver the application service to the end user.

The precise service offered by the application layer will vary from application to application, but will often involve identification of the communicating partners and the agreement of the responsibility for error recovery, security aspects and data encoding. The application layer is also responsible for negotiating and meeting certain quality of service requirements for reliable data transfer, throughput or for delays. Some applications are loss tolerant while others are loss sensitive. Some applications are bandwidth sensitive

[1] *A web browser such as Internet Explorer or Netscape Navigator is an example of a user agent. It provides the interface between an end user and the world wide web application. Email clients are also examples of user agents.*

[2] *Both web browsers and email clients conform to the client server model.*

while others are elastic in their bandwidth requirements. Similarly some applications are delay tolerant while others are delay sensitive. Tanenbaum provides a useful table[3] that describes the quality of service requirements for different network applications.

[3] *Tanenbaum, Figure 5–30, page 397.*

4.2 Interfaces

For user agents, the interface to the application layer is today likely to be a **Graphical User Interface** (**GUI**), such as that provided by the Windows operating system. It could also be a command line interface, as provided by DOS or Unix.

Application layers may also provide services to other applications by means of an **Application Programming Interface** (**API**), which will provide a set of library functions that can be called from application programs.

4.3 Functions

The main function of the application layer is to organise the necessary resources to allow an application process on one system to communicate with an application process on another system via a network. The application layer may also synchronise the application processes at both ends so that they can communicate successfully.

The application layer will often provide reliable communications to the application processes, especially when the application layer makes use of an unreliable transport service. The application layer is the layer of last resort which must correct all the problems that have not been dealt with by the lower layers. The application layer must therefore, unless it is using a reliable transport service, be able to detect the loss, corruption and duplication of messages and be able to recover from these problems. It must also be able to control the flow of data if the receiver or the network cannot handle the rate of data being transmitted.

The fundamental decision that designers of applications must take is which transport service to use. It is the transport service that supports the differing levels of service that might be required. For internet applications, there is a choice between a **reliable service** using the **Transmission Control Protocol (TCP)** or an **unreliable** or **best efforts service** using the **User Datagram Protocol (UDP)**. The reliable service uses a complex connection oriented transport protocol, and the unreliable service uses a simple connectionless transport service. The choice of what sort of transport service to use will have a huge effect on what functions are required in the application layer. It may seem strange at first to think that some application developers would prefer to use an unreliable transport service, but there are several reasons why this might be the best choice.

- Real time applications are more tolerant to packet loss than they are to delay. Losses of occasional packets will not make much difference to audio or video transmissions, as losses are relatively infrequent and the applications can interpolate missing data, so that losses can be hidden from users. Delays, however, and particularly variable delays, do cause problems that can be observed by users. They can result in a jerky effect which is quite disconcerting. Using TCP, as we will discover later, does give rise to extremely variable delays, while delays with UDP are less severe and are more consistent.

- The client server model is well suited to using a connectionless transport service. If clients make occasional request to servers, then using a reliable connection-oriented service, such as that provided by TCP, can be very inefficient. It will be necessary to set up a connection and close it down afterwards. This will require a minimum of five packets and the server will have to hold state information in its memory about all the transport connections that are currently active. Communications operate much more efficiently, as do applications, if a connectionless transport service is used. There will only be a need for two packets to be exchanged and the server can minimise memory usage as it does not need to hold any state information about connections. If packets are lost or corrupted the application client simply retransmits its request.
- If applications are very security conscious, they will not trust the transport service or anything else that was developed or is managed by other parties. The designers of such applications will want to detect and recover from errors within the application itself. In this case, it would be pointless to replicate this functionality in the transport layer and it would be much more efficient to use an unreliable transport protocol.
- The implementation of a reliable transport service requires a large amount of complex code. Simple devices without much memory or processing capability are unlikely to be able to support a reliable transport service. The memory and processing requirement overhead for a reliable transport service may force application designers to choose a connectionless transport service.

Some applications that make use of connection-oriented transport services may also allow the grade of service to be requested and negotiated when the connection is established. The parameters that are of interest relate to reliability, throughput and delay.

4.3.1 Encapsulation

Data from end users or other applications is encapsulated in an application layer PDU by prefixing the data with an application layer header specific to the application protocol.[4]

4.3.2 Addressing

Addressing is often thought of as a function of lower layer protocols, but many application protocols do require their own addressing function as well as having to pass down addresses to be used by lower layer protocols.[5]

4.3.3 Ordered delivery

Ordered delivery is a function of a reliable transport or network service, but where an application is using an unreliable transport service, PDUs can be received in the wrong order as they can take different routes through the network. Where this happens, the application layer protocol must contain a sequence number field so that the application layer can determine if PDUs arrive out of order. The sequence numbers can also be used to request that lost PDUs are retransmitted, and to reorder them if necessary so that they can be handed in order to the appropriate application process or user agent. To do this the application layer needs to buffer the PDUs received, and therefore a certain amount of memory must be allocated for storing incoming PDUs while earlier PDUs are awaited.

[4] For example, an email address message body is encapsulated with Simple Mail Transfer Protocol headers containing fields such as the FROM field and the SUBJECT field.

[5] For example, when requesting a web page using Hyper-Text Transfer Protocol (HTTP), the 'address' of the page will be indicated by the Uniform Resource Locator (URL) field. The URL will indicate the application layer protocol to be used (HTTP), the name of the host from which the web page is to be fetched (network layer addressing), the port number to be used (transport layer addressing) as well as the file to be transferred (application layer addressing).

4.3.4 Flow control

Applications which do not make use of a reliable transport service will also require an end-to-end flow control mechanism so that the receiver can regulate the flow of data from the transmitter. To do this the application protocol header will require a sequence number and an acknowledgement field so that the receiver can acknowledge each PDU transmitted. It can then slow the rate of transmission by not acknowledging PDUs until it is ready to receive some more.

4.3.5 Error control

Where an application is using an unreliable transport service (or where the application does not trust a reliable transport service), the application must perform its own error detection and recovery. This will require a redundant error checking field in the application protocol header as well as sequence number and acknowledgement fields, so that the application layer can check that all the PDUs have been received and so that retransmission can be requested and PDUs re-ordered if necessary.

4.3.6 Connection control

The connection control function is required in all application protocols that make use of a connection-oriented transport service. The application layer must be able to establish connections prior to transmitting data and to close them when there is no more data to be transmitted. The application layer must identify and determine the availability of the application processes which wish to communicate and establish their authority to do so. It must also determine the mode of communication (simplex, half duplex or full duplex). A facility is also required so that connections can be reset or reinitialised to a known state should serious problems be encountered.

4.3.7 Security

Security is often an important function of the application layer, as many applications will assume that all networks are insecure, and application layers sometimes prefer to implement security at this level, rather than make use of security functions within the transport and network layers. Schemes are required to authenticate the parties involved in the communication and to prevent any other parties from being able to read or alter the data being transmitted.

4.3.8 Data encoding

An important function of the application layer is to determine how data is to be encoded for transmission. This may involve the choice of character codes,[6] the use of tags to define how data is displayed[7] or to define data syntax or semantics.[8]

Abstract Syntax Notation 1 (ASN.1) has become an important standard for specifying PDU formats at all layers. It is an ISO standard, but it has also been used extensively by the IETF in specifying new Internet based protocols. It can be thought of as a type definition language where data is defined as belonging to either primitive types such as Boolean, Integer or Bitstring, or more complex user-defined types. It is similar to data type declarations in programming languages. ASN.1 is not only used for specifying protocols

[6] *For example, American Standard Code for Information Interchange (ASCII), Unicode and Extended Binary Coded Decimal Interchange Code (EBCDIC).*

[7] *For example, Hyper-Text Mark-up Language (HTML).*

[8] *For example, Extensible Mark-up Language (XML) and Abstract Syntax Notation 1 (ASN.1).*

(abstract syntax), it is also used to code the data that is being transmitted in the PDUs (transfer syntax). It does this by using a set of Basic Encoding Rules to code the data types and the data.

4.3.9 Data encryption

Data is often encrypted between application layer entities to ensure that it cannot be viewed or altered by third parties as it is transmitted across networks.

4.3.10 Data compression

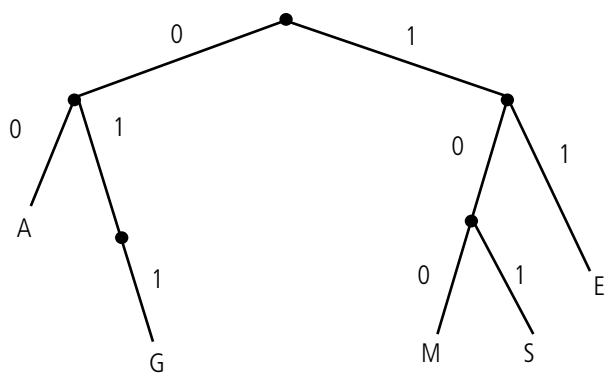
Data compression is often required because bandwidth in wide area networks is a scarce (and hence expensive) resource, and some types of data (such as voice and video) require a large amount of bandwidth, although they use that bandwidth quite inefficiently. Typically voice and video signals (and to some extent also text) contain a large amount of redundant information and can be coded much more efficiently using data compression algorithms. These algorithms can either be lossy,[9] where information cannot faithfully be reproduced at the receiver, or they can be lossless,[10] where the quality of the information after decompression at the receiver is just as good as it was before it was compressed at the transmitter.

There are many different complex data compression algorithms used for coding data prior to transmission to conserve network capacity. All of them apart from Huffman Coding are beyond the scope of this syllabus. Huffman Codes use variable length codes for different symbols depending on how frequently they are used. The ASCII character set uses 8-bit codes (including a parity bit) to define all the characters of the alphabet and other characters. But some characters occur much more frequently than others. Huffman Codes allow more frequently used characters to be represented by fewer bits, and less frequently used characters to be represented by more bits. By doing this a significant reduction can be achieved in transmitting a large amount of text. But, if characters are represented by variable length codes there must be a clear method for determining the start and end of the code for each character. Huffman Codes do this in a clever way by coding each character as the path from the root to a leaf of a binary tree called a Huffman Tree. Thus messages can be encoded in an unambiguous way, so that the receiver can always decode the message and knows that when it reaches a leaf node, a character has been received.

[9] *E.g. Joint Photographic Expert Group (JPEG) compression of still images and Motion Picture Expert Group (MPEG) compression of video.*

[10] *For example, Huffman coding of text.*

Figure 4.1: Example of a Huffman Tree to encode the word MESSAGE



The Huffman Tree in Figure 4.1 can be used to encode letter M by navigating down from the root to the leaf node M and writing down a 0 whenever a left branch is taken and a 1 whenever a right branch is taken. M is therefore encoded as 100. Similarly E is encoded as 11, S as 101, A as 00 and G as 011. The word MESSAGE is therefore encoded as 100111011010001111.

The bit string above can be unambiguously decoded by starting at the root and taking a left branch whenever there is a 0 and a right branch whenever there is a 1 until a leaf is reached which signifies the coded letter. The next letter is decoded by the same method starting again at the root.

A Huffman code, although designed from a tree, can also be defined in a table. The Huffman code defined in the tree in Figure 4.1 could also be defined by means of Table 4.1.

Table 4.1: Example of a Huffman Code table to encode the word MESSAGE

A	00
E	11
G	011
M	100
S	101

From a Huffman Code table, a tree can be drawn by placing each letter as a leaf of the tree according to the navigation rule for the code (left branch for a 0 and right branch for a 1).

For codes involving many letters, it is easier to code from the table and to use the tree for decoding.

4.4 Virtual terminal protocols (Telnet,[11] SSH and VT)

[11] *RFCs 854–861.*

4.4.1 Telnet

Telnet was one of the original Internet application protocols. It is an example of a virtual terminal protocol that allows users on a character-mode terminal (or more commonly these days on a PC running a terminal emulator) to log into and execute commands on a remote host using a command line interface. It uses a reliable TCP connection. Telnet just relays any characters typed by a user to the remote host, and allows commands to be entered at the remote host as if they came from a local terminal. It then relays any characters sent in response back to the user. These characters will include any user name and password requests, but it should be noted that passwords will be transmitted as plaintext and the protocol therefore is not secure. Telnet can support many different terminal types and translate between different character codes, if necessary. It does this by translating to a standard format, known as **Network Virtual Terminal (NVT)** for transmission across the network.

Telnet command codes can be embedded in the data stream. To achieve this, a special escape character (FF in Hexadecimal) is required so that the receiver knows to interpret the next character(s) as a command code. Embedding control information like this within data is known as **in-band signalling**. Also, the user has sometimes to enter a command to Telnet to request the Telnet client to close a connection, for instance, and an escape character (usually CTL-SHIFT-6) is needed so that the next line is interpreted

as a Telnet command to the client and not as data to be sent to the server. This illustrates a problem with in-band signalling, that data transmission will not be completely transparent, as some combinations of characters cannot be sent as they may be interpreted as commands.

Telnet messages do not have application layer headers. They consist of the characters being typed by the user or sent by the server together with occasional command codes generated by the Telnet client or server.

A further potential problem with Telnet that makes it very inefficient over WANs is that it was designed for use over asynchronous modem links. The standard method of remotely accessing a host at the time was for a character to be sent to the host and the host to echo it back to the terminal before it was displayed. This method of operation is known as **echoplexing**. It required a full-duplex link, but had the advantage of showing users when noise on the line was corrupting data. It is not well suited to modern packet-switched networks because there is a delay between typing a character and seeing it appear on the screen, and each character will be transmitted to and from the host in a single packet, incurring large protocol overheads. Also, with modern WAN links, the probability of characters being corrupted is quite low. Because of this, it is advisable to turn off the echo function at the remote server and enable a local echo at the client.

Telnet is not commonly used today because of security concerns, but it is still often used by network managers to remotely manage and configure routers. Network managers can make Telnet much more secure by configuring the routers to only accept Telnet sessions from known IP addresses, corresponding to the network management workstations.

4.4.2 SSH

Another way to improve security for remote login type applications is to use **Secure Shell (SSH)**. The functionality of SSH is very similar to Telnet, but user names and passwords are encrypted for transmission from SSH clients to SSH servers.

4.4.3 VT

Virtual Terminal (VT) is the ISO equivalent protocol that offers similar functionality to Telnet, but like many other ISO protocols, it has found it difficult to compete with the protocols designed for use on the Internet.

Activity 4.1

Attempt to access a host at your university/institution from a PC or another Unix host using Telnet by entering telnet hostname at a DOS/Unix command prompt. It may be that Telnet access to hosts will be barred for security reasons. If this is the case, investigate whether SSH is supported and if so, attempt to establish an SSH connection to a remote host.

Then do the same from a web browser by entering telnet://hostname in the address window of the browser.

4.5 Web protocols (HTTP[12] and HTML)

[12] *RFCs 1945, 2068.*

The world wide web, **Hypertext Mark-up Language (HTML)** and the **Hyper-Text Transfer Protocol (HTTP)** were all invented by Tim Berners-Lee at CERN in 1989. The world wide web is now the most important and widely used Internet application.

HTTP and HTML were studied in a Level 1 unit. You are strongly advised to revise this material, as the remainder of this section on web protocols will assume this knowledge and build upon it.

HTTP assumes a client server model for communications. The web browser acts as the client and requests web pages from the web server. The page is referenced by a **Uniform Resource Locator (URL)** which can be thought of as an application layer address. It defines the protocol to be used, the location of the server and the file to be transferred as well as the transport layer address to be used.

A URL has the following structure:

protocol://hostname/filename:port number[13]

The protocol is usually HTTP (in lower case), but browsers do support other protocols, such as FTP and Telnet. The URL can also be used to access a local file by specifying "file://" as the protocol followed by the full directory path for the file. The protocol field of the URL is always terminated by "://"

The hostname is the domain name of the host and is a series of hierarchical domain names in ascending order separated by ".". By convention, the hostname of most web servers usually start with "www." as the lowest level in the hierarchy of domains. The filename includes a full directory path with directories being delimited by "/"s. Finally, the port number (which is optional) is preceded by ":" and is used to indicate the transport layer address that is to be used. HTTP employs TCP for its transport service and, by default, will use port 80 which is the registered port number for HTTP.

The HTTP protocol is actually quite simple, as there are only a small number of basic operations, called methods. The most common is the GET Request and Response that fetches a web page from a server. At its most basic the request header could just contain a field to indicate the method, the URL of the required web page and the version of HTTP to be used. The response will normally be the requested file, but the response header also contains a status code. Some of these status codes will be displayed by the browser in the event of a failure.[14] In HTTP Version 1.0, a TCP connection was opened for each object to be transferred and closed afterwards. These are known as **non-persistent** connections. In HTTP Version 1.1 and all subsequent versions, **persistent** connections were supported and many objects could be transferred over the same connection, thus improving response times. Response times were further improved in HTTP 1.1 by allowing the client to issue new requests before responses had been received to earlier requests. This facility is known as **pipelining**. The default mode of HTTP 1.1 is to use persistent connections with pipelining.

HTTP has a simple security mechanism that developers can implement to help prevent unauthorised access to web pages. A web page can be set up so that authorisation is required. In this case the web server prompts the client for a user name and password. The server requests a user name and password with a 405 Authorization Required Response and the browser prompts the user for this information. Once the browser has obtained a user name and password it resends the request but this time includes the user name and password in the request header. The server will then check this and if satisfied, will download the page. The browser will cache the user name and password and will automatically include them in any further requests to the server during that browser session.

A secure version of HTTP called **Secure HTTP (HTTPS)** has been implemented which is identical to HTTP but runs over a secure transport connection provided by the Secure Sockets Layer (SSL) which is

[13] For example, http://doc.gold.ac.uk/~mas01pt/cis 222:80 indicates that the file ~mas01pt/cis222/index.html (index.html is the default name of the file to be retrieved) is to be requested using HTTP from the Goldsmith's Department of Computing Web Server on the UK academic network (doc.gold.ac.uk) using port 80.

[14] For example, 200 OK, 301 moved permanently, 400 Bad Request, 401 Authorization Required, 404 Not Found and 505 HTTP Version Not Supported.

implemented between the application layer and the transport layer and certifies the identity of the web server. You can tell when you are using HTTPS because the URL will be commence with https:// in the address window of the browser and a padlock icon will appear at the bottom of the browser window. Details of the certificate can be obtained by double clicking the padlock icon.

Activity 4.2

Attempt to set up a Telnet connection to a web server at your university/institution or another other web server on the Internet using the HTTP port (80). The DOS/UNIX command line syntax for this will be **telnet hostname 80**.

For example, type **telnet websiteaddress 80** to establish an HTTP session with a host web server that you know. Then once the connection is established, type **GET pagename HTTP/1.0.**

The pagename can be index.html for the default home page of the site or it could be a structured hierarchical name. This will fetch the HTML code for the appropriate page from the web server.

4.6 Mail protocols (SMTP[15], MIME[16], POP3[17], IMAP[18] and MOTIS)

[15] *RFC 821.*
[16] *RFCs 2045–2049.*
[17] *RFC 1939.*
[18] *RFC 1730.*

Electronic Mail was also one of the earliest Internet application protocols to be designed. It is a store and forward text messaging protocol supporting mail clients (user agents that send and receive messages) and mail servers that relay messages to each other and to and from mail clients.

Email clients have five basic functions:

- **composition,** which allows users to create messages
- **transfer,** which allows users to transfer messages to and from the mail server
- **reporting,** which allows the mail server to indicate such things as a message not having been delivered
- **displaying,** which allows the mail client to display the headers of messages and their contents
- **disposition,** which allows the user to delete messages or store them in folders.

Email clients use two different protocols. One for sending messages to the mail server and one for retrieving messages from the mail server.

4.6.1 Simple Mail Transfer Protocol (SMTP)

Simple Mail Transfer Protocol (SMTP) is used to transfer messages from a mail client (user agent) to a mail server (mail transfer agent) and is also used to transfer messages between mail servers. It is a very simple text based protocol. Messages comprise a set of headers and a body. There are two envelope headers which start with MAIL FROM:, used to identify the message originator and RCPT TO:, used to indicate the recipient(s) of the message. Unlike many more modern protocols, each envelope header is transmitted and acknowledged separately, rather than encapsulating the body. The message itself is prefixed by the keyword DATA followed by the text to be transmitted, but this also has its own headers such as FROM:, TO:, SUBJECT: and DATE: from which the envelope addresses are obtained. These headers are followed by a blank line and then the actual text of the message. The body is terminated by a new line with just a full stop on it and

then another new line. All bodies and headers are coded in 7-bit ASCII text. SMTP is not a real-time protocol. Messages are stored at clients and servers and forwarded at regular intervals using reliable TCP connections. SMTP PDUs do not really have application layer headers as normally understood. They consist of keywords followed by some data.

SMTP does not offer any guarantees about delivery of messages, although it is quite robust and considered to be reliable.

4.6.2 Multipurpose Internet Mail Extensions (MIME)

Because SMTP was designed only to carry ASCII 7-bit characters, it cannot on its own be used to transfer 8-bit binary data which would be required if an executable file or a formatted text file (such as a Word document) were to be transmitted. When the designers of SMTP realised that users wanted to the ability to send data other than ASCII text, they were faced with two possible solutions. Either they could change the SMTP protocol and update all the clients and servers at the same time, or they could just create a new protocol to allow 8-bit data to be encoded as 7-bit ASCII characters. They chose the latter option, as it only involved upgrading the clients, and they designed a new protocol called **Multipurpose Internet Mail Extensions (MIME)** which allowed 8-bit data files to be attached to SMTP messages and be transmitted as 7-bit ASCII characters.

MIME defines a number of standard data types and sub-types.[19] These MIME data types have become a standard in many Internet and other applications.

[19] *For example, text/plain, text/html, image/jpeg, application/msword, application/pdf, audio/basic and video/mpeg.*

SMTP assumes that mail servers operate continuously and are always available. If they are not available, the messages will be stored and forwarded when the mail server becomes available. Clients, however, are frequently not available, as users do not keep their mail clients running all day and often do not have a permanent connection to the Internet. For this reason SMTP is not well suited for delivering messages to clients. Instead, other protocols were designed to allow clients to connect to servers and request that messages are downloaded. Because these protocols were designed to work over dial-up networks, they also require security mechanisms to ensure that mail is being downloaded by valid users.

4.6.3 Post Office Protocol 3 (POP3)

One protocol that does this is **Post Office Protocol 3 (POP3)**. The protocol has three phases. It has an authentication phase where the user is authenticated by a user name and password; a transaction phase where messages are downloaded from the mail server; and an update phase where the messages on the server will be deleted (if required) after they have been successfully downloaded to the client.

4.6.4 Internet Mail Access Protocol (IMAP)

An alternative protocol for retrieving messages from mail servers is **Internet Mail Access Protocol (IMAP)** which offers very similar functions to POP3, but also allows users to view message headers and select which messages to download. This is very useful for a dial-up connection where bandwidth is expensive and should not be wasted by downloading spam or other unwanted messages. IMAP also allows messages to be stored in folders on the server, which is particularly important if the user often accesses his email from different machines.

4.6.5 Hyper-Text Transfer Protocol (HTTP)

Another, and increasingly popular method for sending and receiving email messages is to use the world wide web to access mail services such as Hotmail, making use of the web's **Hyper-Text Transfer Protocol (HTTP)** rather than using mail protocols. As with IMAP, messages can be organised in folders on the server. Web-based email has the advantage of allowing users to access their email from any machine that supports a web browser, such as a PC in an Internet Café.

4.6.6 Message-Oriented Text Interchange Standard (MOTIS)

Message-Oriented Text Interchange Standard (MOTIS) is an ISO messaging standard. It is based on the ITU-T **X.400 Message Handling Service (MHS)** standard. This standard, unlike SMTP, is very complex and sophisticated and does many things that SMTP does not do. It has not been very successful and virtually all email today still uses SMTP. MOTIS/X.400 is perhaps too complex and users prefer the simplicity of SMTP, particularly with regard to email addresses. X.400 addresses consist of a set of keywords and values which are much harder to remember and more cumbersome to use than SMTP addresses.[20]

[20] *As an X.400 address, p.tarr@gold.ac.uk would appear as, I=P; S=TARR; O=GOLD; PRMD=UK.AC; ADMD=; C=GB.*

Activity 4.3

Attempt to connect to the mail server at your university/institution or your ISP's mail server using the SMTP port (25). The DOS/UNIX command line syntax for this will be **telnet hostname 25**. You should be able to find the domain name for the mail server by looking at the configuration of your mail software. The port number is a transport layer address which will be discussed in the next chapter.

You will now be able to enter the following SMTP commands in **bold** and you should receive the SMTP responses in ordinary type.

MAIL FROM: <your email address enclosed in angle brackets>

250 <your email address enclosed in angle brackets> is syntactically correct

RCPT TO: <your email address enclosed in angle brackets>

250 <your email address enclosed in angle brackets> is syntactically correct

DATA

354 Enter message, ending with "." on a line by itself

From: <your email address>

To: <your email address>

Subject: Test

This is a test.

.

250 OK id=<message id provided by mail server>

QUIT

If you follow the above procedure, your mail server should deliver a message with the subject "Test" to your mailbox. Note that the DATA field which also contains some mail headers is terminated by a line that contains just a full stop. Using Telnet to mimic SMTP works because Telnet allows TCP connections to be set up to any port number (not just on port 23 that Telnet uses by default) and because it allows ASCII characters to be transmitted transparently without any protocol headers. The mail server is also expecting to receive ASCII characters but only on TCP connections using port 25.

4.7 File transfer and access protocols (FTP[21], TFTP[22], NFS[23] and FTAM)

[21] *RFC 959.*
[22] *RFC 1350.*
[23] *RFC 3530.*

4.7.1 File Transfer Protocol (FTP)

File Transfer Protocol (FTP) was also one of the original Internet application protocols. It allows three different types of file (unstructured, structured and random) to be transferred over a network between one host and another using a set of simple commands. File transfers are now quite often carried out using HTTP rather than FTP, but FTP is still used not least when web pages are published to a web server. Most web publishing software uses FTP, although the details of it are hidden from the users and few web designers will be aware that they regularly use FTP.

The original FTP implementations were command line interfaces for use on Unix hosts, and allowed users to view directories on remote hosts and the files they contained, change directories when necessary using Unix commands and then download or upload files between two hosts.[24] FTP converts these user commands to standard FTP commands (three or four letter codes) which are sent via the control connection and elicit responses containing a three digit status code followed by some text from the server. Modern FTP implementation uses a graphical user interface where the local and remote directories can be viewed together and files to be transferred can be highlighted and then transferred by clicking an arrow button that indicates the direction of the transfer.

[24] *Typical FTP user commands include **open** to establish an FTP command connection to a host, **close** to close an FTP command connection, **ls** to list the contents of a remote directory, **cd** to change a remote directory, **get** to open a data connection and transfer a file from the remote host, **put** to open a data connection and transfer a file to the remote host, and **quit** to close FTP*

The protocol actually makes use of two TCP connections: one for control commands and responses, and the other for actual file transfer. The control connection is left open for a whole FTP session, while the data connection is established to transfer a file and closed as soon as the file has been successfully received. This method of control using one connection for control and another for data is called **out-of-band signalling** as opposed to other protocols such as Telnet and HTTP which use **in-band signalling**. Because all control information in FTP is passed via the control connection, the data connection does not require any application layer headers and is a simple TCP connection.

FTP can convert between different character codes. It converts data to the same Network Virtual Terminal (NVT) codes as Telnet for transmission.

Unlike HTTP, FTP cannot be a stateless protocol. The FTP server has to remember which connections belong to which FTP users, and also which current working directories each FTP user is using.

The connection used for commands actually uses the Telnet protocol to transfer commands and responses to the remote host. For security purposes, hosts normally require a user name and password to be entered for all Telnet sessions. But this would hinder general public FTP access to a site, as users would have to pre-register. A convention that developed for public FTP access was for public FTP servers to accept the user name 'anonymous', but not to perform a password check. Instead, anonymous users are expected to enter their email address as the password, so that the host can, if it wants, collect some details on the users of its public FTP service.

4.7.2 Trivial File Transfer Protocol

There are some situations where a complex file transfer protocol is inappropriate. Some network devices, such as low-cost routers, do not have sufficient memory and processing capability to justify the implementation of

such a protocol, if it has to sit on top of a complex connection-oriented transport protocol. Also, between two hosts on the same LAN the probability of errors is quite remote and it may be preferable to use a lighter weight file transfer protocol. A disc-less work station, which has to download all its software from a server over a LAN, is a good example of a situation where only a light-weight file transfer protocol is desirable. **Trivial File Transfer Protocol (TFTP)** is such a protocol. As its name implies, it is extremely simple. It uses UDP as its transport service and provides quite a thin application layer. Each application layer message has to be explicitly acknowledged before another message can be sent. Each TFTP message carrying data contains an application header that includes a sequential block number. The receiver will then acknowledge the receipt of the message with an acknowledgement message that contains the block number. The transmitter will re-transmit a message if an acknowledgement is not received before a timeout expires. Unlike FTP, TFTP only supports file transfer. It does not support any interaction to locate files in directories. The files and the direction of transfer are specified in the command line that is used to call TFTP. TFTP also has no facility for authenticating users. For security purposes, network managers should therefore only allow TFTP traffic to and from known IP addresses. TFTP is often used by network managers to download or upload router configurations and software.

4.7.3 Network File System (NFS)

Sometimes it will be more efficient to access files remotely rather than to transfer them in their entirety. A popular means of doing this is to use the **Network File System (NFS)**, originally developed by Sun Microsystems for the Unix environment, but which has since been ported to most other commonly used environments. NFS allows physically remote directories to be mounted on local systems, so that the directories and their files appear to be local to the users. All the standard operations that are carried out by the Operating Systems on local directories and files are supported transparently on the remote directories and files. NFS is implemented using an application mechanism called **Remote Procedure Calls** (**RPCs**), also developed by Sun, where software that normally calls procedures on the local system can call equivalent procedures on the remote system. RPCs are implemented using a very simple protocol that packs the name of the procedure and any parameters required into a message using a coding system called **External Data Representation** (**XDR**). This is sent as a request to the remote system which unpacks it and calls the procedure. It then packs the return value and any other output parameters into a response message to be sent back to the calling system, which then returns these to the original calling process. NFS can use either TCP or UDP for its transport service but, because it is a simple client server application, it is best implemented on top of UDP. NFS is not very secure, but authentication services have been developed that offer improvements in this area.

File Transfer Access and Management (**FTAM**) is the equivalent ISO protocol to FTP. It is not in common use today.

Activity 4.4

Use the file transfer protocol on a PC or on a Unix system at your university/institution to transfer files to and from an FTP server at your university by entering ftp hostname at a DOS/Unix command prompt, or by using a windows-based version of FTP. The domain name for FTP servers begins, by convention, with "ftp." just as the domain name for a web server, by convention, often begins with "www.". Use the same software to perform an anonymous file transfer from a public FTP server by downloading some RFCs from

ftp://ftp.rfc-editor.org. Then transfer files from the same FTP servers using a web browser by entering ftp://hostname in the address window of the browser.

4.8 Directory protocols (DNS[25], X.500 and LDAP[26])

[25] RFC1034, 1035.

[26] RFC 3377.

4.8.1 Domain Name System (DNS)

Access to directories is required by a number of network functions, as well as by other applications. A directory service called the **Domain Name System (DNS**) was developed for the Internet to allow applications to use host names, and then for these host names to be translated (or resolved) into network layer addresses. In the early days of the Internet, the mapping between flat host names and network layer addresses was done via a text file that was centrally managed and then distributed to all the hosts on the Internet. This method soon became too unwieldy as the Internet grew in size, and an alternative method using a hierarchical, fully distributed system was devised called the Domain Name System.

The DNS required that all host names were hierarchically structured. It can be viewed as a tree with a single root. The top layer of the hierarchy is the last field in the hierarchical name. It can be .com, .org, .edu, .net, etc. or it can be a country code such as .in, .hk, .si, .uk, etc. These domains are allocated by the Internet Corporation for Assigned Names and Numbers (ICANN) which is responsible for managing the thirteen top DNS root servers. For each of the top level domains there are at least two DNS servers. Below this level there are second level domains which manage the authoritative DNS servers for the domain, and have delegated authority to manage all the domains levels below them. They might well delegate further levels to other organisations. This delegation can continue to the third or sometimes the fourth level.[27]

[27] For instance, the host name doc.gold.ac.uk has the UK as its top level domain; the Academic Community is second level domain; Goldsmiths as its third level domain; and the Department of Computing as its fourth level domain. ICANN has delegated the authority for allocating all .uk domain names to the UK registration authority (Nominet) which has delegated authority for allocating .ac.uk and .gov.uk domain names to the UK Education and Research Networking Association (UKERNA) who have allocated the gold.ac.uk name to Goldsmiths College who have then allocated the doc.gold.ac.uk domain to the Department of Computing.

When an application needs to resolve a hostname it calls a name resolving function to issue a DNS request to its local DNS server. The network address of the local DNS server is configured in the client computer. The local DNS Server will hold data on all domains for which it is authoritative as well as holding a cache of previously resolved names. If the local server is able to resolve the name, it will return the network layer address in a DNS response. If the server is not able to resolve the name, it will make a request to one of the root servers. The root server will return the correct network layer address if the name is held in its own data or cache, otherwise it will return the network layer address of an authoritative server that can help resolve the name. This process could repeat itself a number of times until either the name is resolved or it can be firmly established that the name cannot be resolved. This process can be either recursive, where the request is passed on from server to server and the responses passed back along the chain of servers, or iterative, where a server makes a number of requests in turn to different servers. DNS can operate over either UDP or TCP transport protocols. Because it is a very simple client server application, virtually all implementations of DNS use UDP for efficiency reasons.

DNS is also used by Internet mail servers to resolve the domain name part of email addresses to the network layer address of the destination mail server.

4.8.2 X.500

The ITU-T and ISO have also developed a comprehensive and complex directory service and protocols. Their directory system is often known by its ITU designation X.500. It operates between Directory User Agents (DUAs)

and Directory Service Agents (DSAs). The protocols used are the **Directory Access Protocol (DAP)** which is used between DUAs and DSAs and **Directory Service Protocol (DSP)** used between DSAs. X.500 like X.400 is based on an object-oriented design and it also incorporates ASN.1 coding. Although X.500 was originally designed as the directory service for resolving names into X.400 mail addresses (a much needed service given the unfriendliness of X.400 mail addresses), it was designed to be much more generic than DNS and can support directories for any type of object stored in a Directory Information Base (DIB). DAP and DSP make use of a number of different ISO protocols in each layer of the ISO protocol stack.

4.8.3 Light-weight Directory Access Protocol (LDAP)

The X.500 directory service, like most other ISO protocol developments, could not compete with its Internet equivalent. However, the design and many of the ideas behind X.500 are good and there is a need for a well-designed directory service that companies can use to look up employees' phone numbers, email addresses and office locations. With this in mind, the University of Michigan adapted X.500 to create a less complex protocol called **Light-weight Directory Access Protocol** (**LDAP**), which, despite its name, is still not particularly light-weight. LDAP is used in Microsoft's Active Directory and Novell's Netware Directory Service products.

Activity 4.5

Try to find a copy of a program called nslookup (Name Server Look-up). It can be found on most Unix systems, but is not usually included with Windows.

Use nslookup to resolve some hostnames. In Unix this can be done on a single command line by entering nslookup hostname at a Unix command prompt. If the name can be resolved, the name and network address of the name server will be displayed, possibly followed by a line that says "Non-authoritative answer:". This means that the name was found in cache. Finally, the name and network address of the host are displayed. If the hostname you look up is obscure enough, you might get an authoritative response that does not include the "Non-authoritative answer:" line.

It is more interesting to run nslookup in debug mode. This can be done by entering the nslookup command without any hostname and then typing set debug before entering a host name on a new line. You will now see more information on the authoritative servers for the host's domain.

If you want to see a verbose account of everything nslookup is doing, there is a second debug mode you can enter by typing set d2.

There are many other options supported by nslookup. You can read about them by typing man nslookup at the Unix command prompt.

You can return from nslookup to the Unix command prompt by typing exit.

An alternative Unix tool for name resolution and DNS debugging is dig (Domain Information Groper). Read about dig and its options by typing man dig and then try to use it to resolve some host names.

4.9 Network Management Protocols (SNMP[28] and CMIP)

[28] *RFC 3416.*

Network management protocols is another area where there was a battle between Internet and ISO standards which was conclusively won by the Internet standards. Yet again the Internet developers chose to launch a simple but effective network management protocol which could be implemented quickly, while the ISO developers created a technically complex protocol that had many more functions and would take a long time to implement and debug.

The Internet Network Management standard is known as **Simple Network Management Protocol (SNMP)** and the ISO standard is known as **Common Management Information Protocol (CMIP)**.

SNMP runs on top of UDP, as it only has to support a small number of request/response interactions to network devices that often have limited memory and processing capability. The protocol does however use the ISO's ASN.1 transfer syntax for encoding data. It operates between a network management station and a management agent within the network device that collects management information in a Management Information Base (MIB). The network management station can request the value of any object in the MIB and can also set the value of an object in the MIB. The network device can also issue a trap to notify the network management station that a problem has occurred.

SNMP version 2 is not very secure as passwords (known as community strings) are transmitted as plaintext. SNMP version 3 addresses these security problems.

CMIP is a complex protocol that makes use of a number of different ISO protocols in each layer and, as a result, is not particularly well suited to managing simple devices which would require considerable memory and processing power to support all the protocols. It does have good security features.

Network management protocols and network management in general will be studied in more depth in the second half of this course.

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
    i. Post Office Protocol 3 allows users to view email headers without downloading the messages' bodies and is the recommended protocol for use with low speed dial-up Internet access.
    ii. Telnet runs efficiently on top of UDP when used with asynchronous character-mode terminals.
    iii. FTP requires that two transport connections are set up in order to transfer a file.
    iv. Simple Network Management Protocol, which was standardised by the IETF, encodes data in ASN.1 which was standardised by ISO.

b. Describe the format of a Uniform Resource Locator.

c. Using the Huffman Code defined in Table 4.1, code the word GAMES. Draw the Huffman Tree and describe how the Huffman code for GAMES can be unambiguously decoded from the Huffman tree.

d. What does the acronym MIME stand for? Why is it necessary for all mail agents to implement it?

e. Describe how DNS resolves a host name into an IP address.

Learning outcomes

At the end of this chapter and the relevant reading, you should be able to:

- describe the services and interfaces offered by the application layer
- describe the functions of the application layer
- identify the reasons why application developers might choose to use an unreliable transport service and how this choice will affect the design of the application layer protocol
- outline how ASN.1 is used to encode data
- encode and decode compressed messages using Huffman codes and trees
- describe how Mail protocols transmit and receive emails
- identify the main differences between POP3 and IMAP in retrieving mail from servers
- describe the format of URLs
- outline how the Telnet protocol works
- outline how security features can be implemented with HTTP
- describe how FTP and TFTP transfer files work and the circumstances in which they are most appropriate
- describe how remote files are accessed through NFS
- describe how DNS resolves hostnames to network addresses
- outline how SNMP manages remote network devices.

Appendix B

Model answers and hints

Chapter 2

a. True or False

   i. TRUE.

   ii. FALSE – they contain virtual circuit numbers instead.

   iii. FALSE – this is transmission delay.

   iv. TRUE.

b. See Section 2.3.2.

c. See final bullet point in Section 2.3.1.

d. Noise Power = 0.2 mW

   Signal Power = 200 mW

   SNR = 200/0.2 = 1,000

   = $10 \log_{10}(1{,}000)$ dB

   = 10 x 3 dB

   = 30 dB

The noise will be amplified by the same amount as the signal (40 dB)

   Let N = amplified noise

   Noise Gain = N/0.2

   = $10 \log_{10}(N/0.2)$ dB

40 dB = $10 \log_{10}(N/0.2)$ dB

40 = $10 \log_{10}(N/0.2)$

   $\log_{10}(N/0.2) = 4$

   $N/0.2 = 10^4$

   N = 0.2 x 10,000

   = 2,000 W

e. Least number of links is a 'daisy chain' topology with five links.

The best way to improve resilience by adding one link is to form a ring.

Maximum resilience will be obtained from a full mesh.

Minimising delay to one node requires a star network centred at that node.

Chapter 3

a. True or False

   i. FALSE – corresponds to top three layers of ISO Reference Model.

   ii. TRUE.

   iii. TRUE.

   iv. FALSE – Service Data Units are passed.

b. See bullet points in Section 3.1.

c. Give three standards body names from Section 3.3 and find an example of

a standard from each (**Note**: IETF = Internet, ITU-T = Telecomms, ISO = miscellaneous).

d. See third bullet point in Section 5.3.

e. See first paragraph of Section 5.4 and Figure 3.5.

Chapter 4

a. True or False

v. FALSE – this is IMAP.

vi. FALSE – it is inefficient and it runs over TCP.

vii. TRUE.

viii. TRUE.

b. See Section 4.5.

c. GAMES = 0110010011101.

d. See Section 4.6.2.

e. See Section 4.8.1.

Chapter 5

a. True or False

i. TRUE.

ii. FALSE – it is initialised to a number derived from the system clock.

iii. FALSE – it is the next sequence number it expects to receive.

iv. TRUE.

b. See first bullet point list of Section 5.2.

c. 10101100 10010101

00100100 11100100

10001000 01110001 = checksum

10101100 10011001 (errors in bits 3 and 4)

00100100 11100100

10001000 01110001

00000000 00001100

As total is not all 0s, errors have occurred.

d. See Section 5.4.

e. See Section 5.5.

Chapter 6

a. True or False

i. TRUE.

ii. FALSE – it's the MTU Size.

iii. FALSE – it's Class B addresses.

iv. TRUE.

b. See Sections 6.1.1 and 6.1.2.

c. 192.16.128.0 = 1100000 00010000 10000|000 00000000

192.16.135.255 = 1100000 00010000 10000|111 11111111

netmask = 1111111 11111111 11111|000 00000000