UNIVERSITY OF LONDON

Data communications and enterprise networking Volume 1

P. Tarr

CO2222

2005

Undergraduate study in
Computing and related programmes

Goldsmiths
UNIVERSITY OF LONDON

This guide was prepared for the University of London by:

P. Tarr

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

london.ac.uk

Published by: University of London

© University of London 2005

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.

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

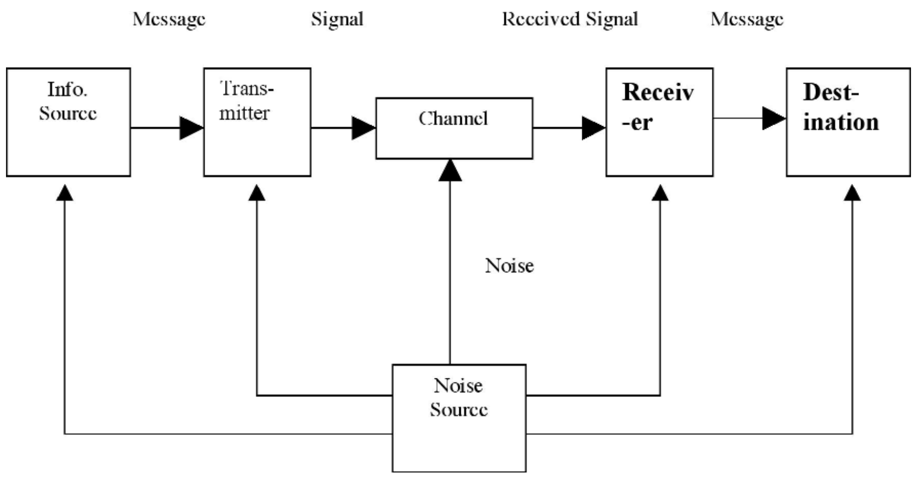
A **signal** is a means of transmitting an element of data across a distance and the process of doing this is called **signalling**. Signals can vary continuously (**analogue signals**) or have discrete levels (**digital signals**). As we will see in Chapter 8, there are four different combinations of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

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:

  $\text{SNR} = 10 \log_{10} (S/N)$, 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.

[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).

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

[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

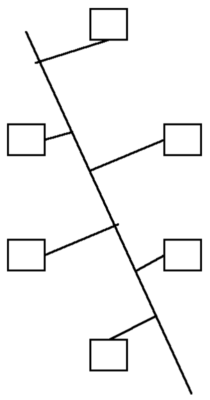


Figure 2.3: Ring Topology

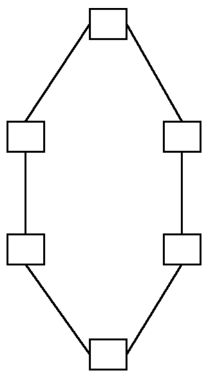


Figure 2.4 Star Topology

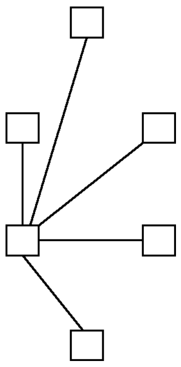


Figure 2.5 Tree Topology

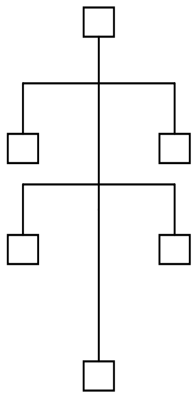


Figure 2.6 Full Mesh Topology

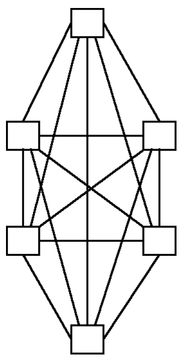


Figure 2.7 Partial Mesh Topology

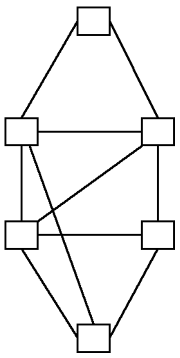


Figure 2.8: Star-Bus Topology

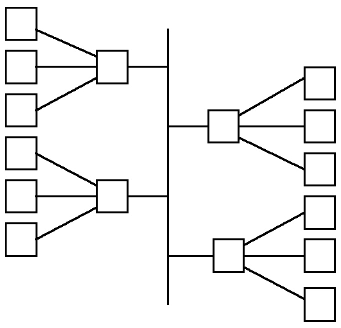


Figure 2.9 Star-Star Topology

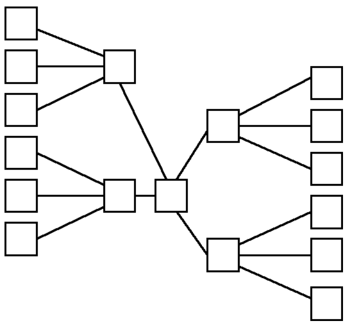


Figure 2.10: Star-Ring Topology

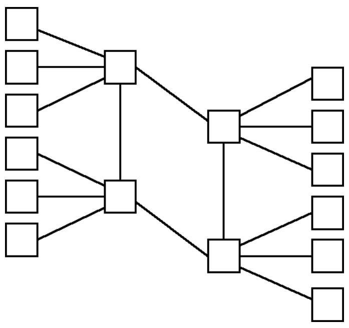


Figure 2.11: Star-Mesh Topology

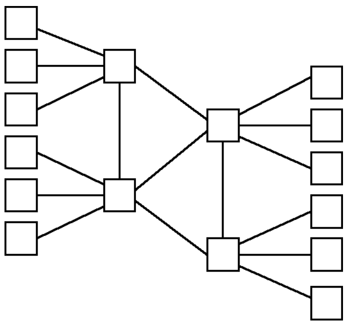
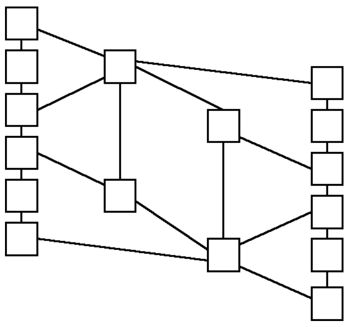


Figure 2.12: Ring-Ring networks



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, EIAITIA 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

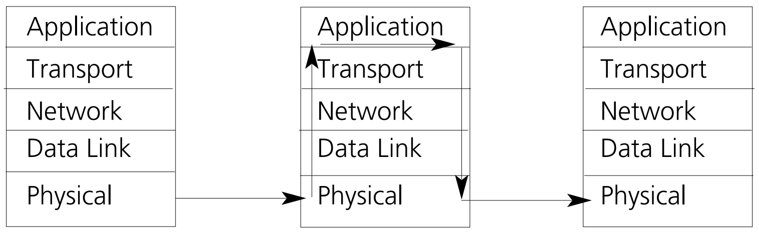


Figure 3.2: A network layer intermediate device between two hosts

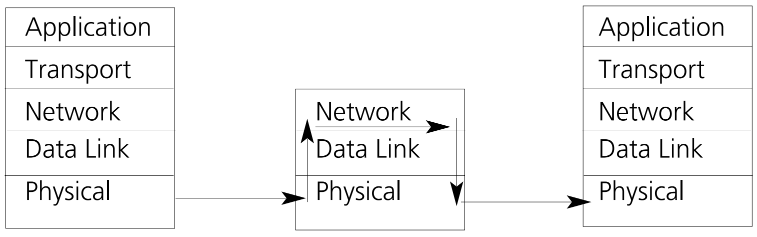


Figure 3.3: A data link layer intermediate device between two hosts

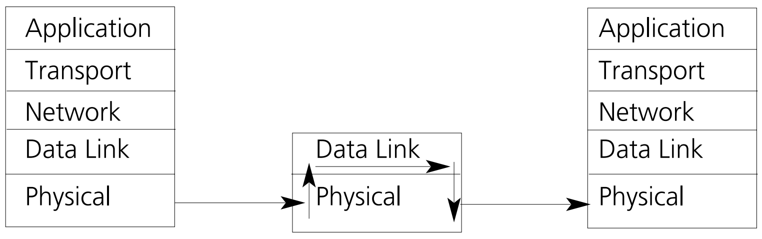
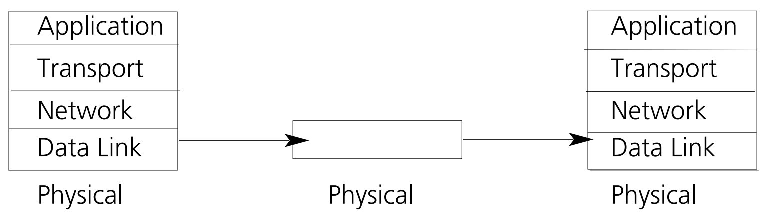


Figure 3.4: A physical layer intermediate device between two hosts



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

		Layer N + 1 Data	Layer N + 1 SDU
	Layer N + 1 Header	Layer N + 1 Data	Layer N + 1 PDU/ Layer N SDU
Layer N Header	Layer N + 1 Header	Layer N + 1 Data	Layer N PDU / Layer N - 1 SDU

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

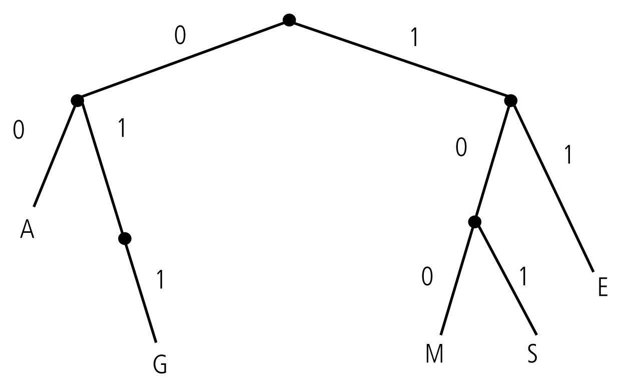
Data compression is often required because bandwidth in wide area networks is a scarce (and hence expensive) resource, and some types of data (such as voice and video) require a large amount of bandwidth, although they use that bandwidth quite inefficiently. Typically voice and video signals (and to 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.

[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.*

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.

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.

Notes

Chapter 5: The transport layer

Introduction

In this chapter, we shall examine the services and interfaces provided by the transport layer and the general functions of transport layer protocols.

Unlike application layer protocols, there are only a limited number of transport layer protocols that can be used. ISO attempted to define four different classes of transport protocols, but none of these has been widely implemented, as the Internet and its own transport protocols gained popularity. In fact two of the ISO transport protocols are strikingly similar to the two main Internet transport protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). We will therefore focus on TCP and UDP in this chapter.

Finally, we shall look briefly at Novell's Sequenced Packet Exchange (SPX) transport protocol and the Transport Layer Security (TLS), which is a transport layer protocol that sits immediately above TCP.

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, (2003). Chapter 17.

Kurose and Ross *Computer Networking – A Top Down Approach featuring the Internet* (Addison Wesley), third edition (2005). Chapter 3.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition (2003). Chapter 20.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002). Chapter 6.

5.1 Services

The transport layer is the lowest layer that truly works end-to-end between application processes. It aims to provide an efficient and consistent service to applications that is independent of the underlying network technologies. In general, it therefore only needs to be implemented on host machines and is not normally implemented on any intermediate network devices.

There are two basic types of transport service that the transport layer can offer to applications. These are:

- A **connection-oriented transport service** which requires connections to be set up before data is transferred and cleared down afterwards and which provides a **reliable** service that guarantees data is delivered in order, without loss and without duplication. A connection-oriented transport service therefore has three phases: one for connection establishment; one for data transfer; and one for connection release. The connection-oriented transport service will make up for any deficiencies in the underlying network services and will improve their reliability.
- A **connectionless transport service** that does not require any connections to be set up before data can be transferred, and provides an **unreliable service** that offers no guarantees regarding data delivery but simply makes its best effort to deliver data. PDUs can therefore be lost, arrive out of order or a duplicate could be received, and the transport

layer will rely on the application layer to detect and recover from these problems, if necessary. A connectionless transport service has only a single phase for data transfer and each PDU is completely self-contained and handled independently of others.

On the Internet, these two basic transport services correspond to the two main transport protocols:

- **Transmission Control Protocol (TCP)** which provides a reliable connection-oriented transport service
- **User Datagram Protocol (UDP)** which provides an unreliable connectionless transport service.

5.2 Interfaces

The transport layer can be implemented in a number of different ways which will influence the interface it offers to applications.

- The transport layer could be implemented as a separate user process, in which case the interface to applications would be via the inter-process communications supported by the operating system.
- The transport layer could be implemented within the operating system kernel, in which case the interface would be via a call to the operating system.
- The transport layer could be implemented in a software library, in which case the interface would be via a call to a library function.

The latter two types of implementation are the most common. The Berkeley Unix operating system (and its derivatives) implements the transport layer within the Unix kernel using socket interfaces. A socket appears to users like a file descriptor which can then be read from and written to in a similar way to a file. Applications create and open sockets and bind them to socket addresses which can be a Unix file pathname for internal inter-process communications, or a combination of network address and transport address used for external communications.[1]

[1] *For external communications Unix will bind the socket with an address comprising an IP address and a transport layer port number.*

Windows implements its transport layer interface by means of calls to functions in a dynamic link library called Winsock.dll which supports a subset of the Berkeley socket calls with some minor changes.

The main primitives used with connectionless sockets are:

- socket() to create a new socket
- bind() to bind a socket with an address
- recvfrom() to receive data
- sendto() to send data.

The main primitives used with connection-oriented sockets are:

- socket() to create a new socket
- bind() to bind a socket with an address
- listen() for servers to specify a queue for incoming connection requests
- accept() for servers to wait for a connection request
- connect() for clients to issue a connection request
- recv() to receive data from a connection
- send() to transmit data over a connection
- close() to close a connection.

5.3 Functions

5.3.1 Encapsulation

The transport layer encapsulates application layer PDUs with a transport layer header to form a transport layer PDU. At the receiving end, the transport layer will process the transport layer header, decapsulate the application layer PDU and pass this to the application layer.

5.3.2 Addressing

The transport layer must be able to identify the application processes with which it is communicating. It does this by allocating a transport layer address to each application process.

On the Internet, both TCP and UDP use a common addressing scheme. This is based on 16 bit port numbers that identify applications. You will have already come across these port numbers if you completed the activities in the last chapter, as they were used to allow Telnet to mimic SMTP and HTTP clients.

Port numbers are controlled by the **Internet Assigned Numbers Authority (IANA)** which publishes all assigned port numbers in a text file on its web site.[2]

[2] *http://www.iana.org/assignments/port-numbers*

The port numbers can belong to one of three ranges.

Ports **1-1023** are the **well-known port numbers**. These are used to identify the main Internet applications and are allocated by IANA. On most systems, they can only be used by system (or root) processes executed by privileged users.

Ports **1024-49151** are the **registered port numbers**. These are registered by developers with IANA. On most systems, they can be used by ordinary user processes.

Ports **49152-65535** are the **dynamic or private port numbers**. These are not registered with IANA and are used for private purposes or are allocated dynamically to clients of applications.

Servers will normally use the port numbers appropriate to each application out of the well-known or registered port number ranges. Clients will normally be allocated port numbers dynamically by their operating systems so that no two client processes on the same host are allocated the same port number.

Activity 5.1

Visit the IANA Port Registry at http://www.iana.org/assignments/port-numbers and scroll down the text file. Firstly note that there are many port numbers registered. See how many applications you recognise and note their port numbers.

Table 5.1 below shows some commonly used well-known port numbers for applications that were discussed in the last chapter.

Table 5.1: Some well-known port numbers

Application	Port No	Application	Port No
FTP Data	20	HTTP	80
FTP Control	21	POP3	110
SSH	22	Sun RPC	111
Telnet	23	IMAP	143
SMTP	25	SNMP	151
DNS	53	LDAP	389
TFTP	69	HTTPS	443

5.3.3 Multiplexing

The transport layer can multiplex data between different applications by using transport layer addresses (port numbers). The destination transport layer addresses will be a field in the transport PDU header and the receiving transport layer will be able to use this to determine to which application process it must forward the application PDU. The transport layer can therefore multiplex data between different applications residing on the same source and destination hosts.

Activity 5.2

Enter the command **netstat –a** at a DOS/Unix command prompt to check what ports on your computer are currently in use and which are used by transport protocols to multiplex data received to different applications. The **–a** option will ensure that all ports are listed. If you are issuing this command on a server, the output is likely to be large, and it is advisable to pipe it through more by typing **netstat –a | more**. Look through the output and identify the client applications that are using dynamic port numbers and server applications that are using well-known or registered port numbers. Note whether the application uses TCP or UDP as its transport protocol. The netstat output will display the local hostnames and port numbers, separated by a colon, followed by the remote hostnames and port numbers. If netstat knows the port number, it will replace the number by some text that indicates the application. If you wish to discover the port numbers used in these cases, you will need to use the **–n** option. If there are any well-known or registered port numbers appearing in the output that you do not recognise, you should check the IANA port registry (URL given in Activity 5.1) to discover what they might be, but remember that application developers are not bound to register port numbers with IANA.

5.3.4 Ordered delivery

Ordered delivery is only guaranteed by a reliable transport protocol. The function relies on sequence numbers in the transport layer header to detect lost, out of order and duplicated PDUs. The protocol mechanisms ensure that data is buffered before being passed on to the receiving application process, and that any data received out of order is re-sequenced and any missing data is re-transmitted, and that the data forwarded to the receiving application process is exactly what was transmitted.

5.3.5 Flow control

Again, this function is only applicable to connection-oriented transport protocols. If the receiver cannot process connectionless transport PDUs quickly enough it is not able to slow down the transmitter. The only course of action is to discard some of the PDUs and rely on the application protocol to detect and recover, if required, from the loss of data.

Connection-oriented transport implementations will have a mechanism to acknowledge the receipt of transport PDUs, and by being slow to issue acknowledgments they will be able to regulate transmission. This mechanism will make use of sequence numbers in the transport layer headers.

Reliable transport layer and many other protocols often use a mechanism for flow control called a sliding window. This mechanism enables the throughput of data on the transport connection to be improved by allowing several PDUs to be transmitted before an acknowledgement is received. The maximum number of unacknowledged PDUs that may be transmitted on the connection is known as the window size. Both the receiver and the transmitter keep track of the sequence numbers of PDUs and acknowledgements sent and received, and ensure that the sequence numbers lie within the range allowed by the window size. TCP uses a slightly different windowing mechanism as it uses the number of characters rather than the number of PDUs to determine the window size.

5.3.6 Error control

For a reliable transport service the transport protocol will perform error detection and recovery. Transport layer headers will contain a checksum field to detect errors. Error recovery is usually done by not acknowledging a PDU in which an error is discovered, thus requiring the transmitter to resend the PDU after a timeout.

Even unreliable transport protocols such as UDP can have an error control function in that there is a checksum field that protects the header and data. However, an unreliable transport protocol such as UDP will not have any facility to make the transmitting transport layer aware that an error has occurred and will just discard the datagram. It will rely on the application layer to detect that a datagram has been lost and to retransmit it, if necessary.

TCP and UDP use the same error detection mechanism. It is called the Internet checksum.[3] It works by treating the data to be protected as a sequence of 16 bit words, padding the data with 0s, if necessary, to ensure that there are a whole number of complete 16 bit words. It then adds all the words together using one's complement binary arithmetic. If, however, there is a carry from adding the leftmost column, then the carry bit is added back in to the rightmost column. This is done to improve the probability of detecting an error in the leftmost column. The one's complement of the sum is then taken and this becomes the checksum.

[3] RFCs 1071, 1141, 1624.

The receiver follows a similar procedure but adds together all the data together with the checksum it received. If the sum is all ones, then there is no error. If the sum contains at least one zero anywhere, then an error has occurred.

In the example below two 16 bit words are added using the above rules to calculate the checksum. At the receiver, two checksum calculations are shown. The first example shows what happens when the data is received without any errors. The second shows what happens when an error occurs in the fifth bit position reading from the from the right.

Table 5.2: Internet checksum example

Sent	Received	
1001 1011 0100 1001	1001 1011 0100 1001	
1110 1011 1101 0111	1110 1011 1101 0111	
1 1000 0111 0010 0000	0111 1000 1101 1110	
1	1 1111 1111 1111 1110	
1000 0111 0010 0001	1	
	1111 1111 1111 1111	**Correct**
Checksum =		
0111 1000 1101 1110		
	1001 1011 0101 1001	
	1110 1011 1101 0111	
	0111 1000 1101 1110	
	0 0000 0000 0000 1110	
	0	
	0000 0000 0000 1110	**Error**

Activity 5.3

Write a program that reads a line of ASCII characters, pads it out with zeros, if necessary, to make a whole number of 16 bit words and then calculates and prints the Internet checksum in hexadecimal format, using the above checksum generation algorithm.

Write another program that reads a line of ASCII characters followed by a 16 bit checksum in hexadecimal format and then verifies the validity of the data by performing an Internet checksum calculation.

5.3.7 Connection control

Connection control is a function that is only required by connection-oriented transport protocols. The transport protocol must have a number of different states in order to establish connections for data transfer and close them afterwards.

Connectionless transport protocols do not require any connection control functions or states, they are able to transfer data without any connection being set up.

5.3.8 Security

Security can be implemented in any layer. The most common layers to implement it are the application layer, the transport layer and the network layer.

There are four requirements for message security:

- privacy – ensures that a message can only be read by the intended recipient
- authentication – ensures that a message comes from the originator who claims to have sent it
- integrity – ensures that a message has not been altered
- non-repudiation – ensures that the message originator cannot subsequently deny that the message was sent.

Security in the transport layer is implemented through Transport Layer Security (TLS) based on the Secure Sockets Layer (SSL) protocol.

5.4 User Datagram Protocol (UDP)[4]

[4] *RFC 768.*

The User Datagram Protocol is a light-weight connectionless and unreliable transport protocol. It does not guarantee the delivery of data in order, without loss or duplication. It simply makes its best effort. As its name implies, it transmits datagrams, which are packets that contain complete addressing information within their headers, so that each datagram can be routed independently. If errors are detected by the receiver or it cannot cope with the rate at which it is receiving data, it will simply discard datagrams.

Describing the protocol as light-weight means that its headers are quite short. In fact the UDP header consists of just four fields with a total of only eight bytes. The first two fields define the **source** and **destination port numbers** (two bytes each) which we have already discussed. UDP uses the destination port number to demultiplex the data it receives to the correct application process. These fields are followed by a **length** field of two bytes which indicates the length of the datagram (including the header) in bytes. Therefore the maximum size datagram that UDP can support is 64 Kbytes. The final field is the Internet **checksum**, which we have already discussed, and which protects the entire UDP datagram including the header together with some fields from the IP Header to form a pseudo-header. The IP fields in the pseudo-header are the source and destination IP addresses, the protocol and the length field. By including these fields within the checksum calculation, UDP protects itself from problems that might occur in IP, such as delivery to the wrong IP address, but it does violate the independence of the layers.

Some applications, such as real-time voice or video, do not require any error checking of datagrams. They can easily cope with errors in the data and there is no point in knowing whether errors exist or not, as they cannot afford to wait for data to be retransmitted. In such circumstances the checksum field is not computed and is set to all 0s. Note that a computed checksum will never be all 0s. In one's complement arithmetic, the number 0 is ambiguously represented by all 0s (+0) or all 1s (-0) and a computed checksum that evaluates to zero is represented by all 1s.

5.5 Transmission Control Protocol (TCP)[5]

[5] *RFCs 793, 1323, 2018, 3390.*

TCP is a complex, full-duplex, point-to-point, reliable, connection-oriented transport protocol. It is a stream-oriented protocol which means that it transmits a stream of characters and uses positions of characters in the stream for the identification and acknowledgement of data segments. TCP receives data from application processes as a stream of data with no message boundaries. The protocol segments the data for transmission and at the receiving end, reassembles it into a character stream for delivery to the receiving application process.

As a reliable protocol, TCP will detect lost, damaged or duplicated segments and will recover from these problems, so that it can guarantee that all segments are received in order, without loss and without duplication. TCP was designed to be used immediately above the unreliable IP protocol to provide a reliable end-to-end service.

5.5.1 Encapsulation

TCP receives data from the application layer and encapsulates the data with a TCP header.

TCP protocol headers are quite long and complex. They are of variable length, depending on which option headers are used. Typically the options header is empty and the TCP header is then 20 bytes long.

The TCP header has the 16-bit same **source** and **destination port number** fields as UDP, followed by 32-bit **sequence number** and **acknowledgement number** fields. The sequence number field is not incremented by one in successive segments, but indicates the position of the first byte of the segment within the data stream. Sequence numbers are not initialised to 0 or 1 when a connection is opened. Instead, when a connection is opened, each end chooses its own initial sequence numbers, usually derived from the system clock. The acknowledgement number is used by a receiver to indicate the next sequence number it is expecting and by doing this it acknowledges receipt of all the characters in the stream up to this point. The next field is an 4-bit **header length** field which specifies the length of the header in 32-bit words. After this field are six unused bits, followed by six **flag bits**. The flag bits are used as follows.

- **URG** indicates that the sending application layer has indicated that there is urgent data (indicated by the **urgent data pointer** later in the header) in the segment that the receiving application layer should process ahead of other data. This flag is rarely used, but it could for instance be used to send an interrupt character to an application process.
- **ACK** is used to indicate that the segment is carrying an acknowledgement. As all data segments will carry acknowledgements, its main purpose is to distinguish between a connection, reset or close request (=0) and a connection, reset or close response (=1).
- **PSH** indicates that the receiver should pass the segment immediately to its application layer and not to buffer it. Again, this flag is rarely used.
- **RST** is used to request that the connection is reset (reinitialised following a problem) and together with an ACK as a response to this request. It is also set when a connection is refused.
- **SYN** is used to request that the connection is opened and together with an ACK as a response to this request. It is short for synchronisation.
- **FIN** is used to request that the connection is closed and together with an ACK as a response to this request.

The next field is the a 16-bit **window** field which is used by the receiver to indicate to the transmitter how many bytes it is allowed to transmit without any further acknowledgements. It is used to indicate the buffer space available in the receiver and to control the rate of transmission via a sliding window mechanism.

The next field is the 16-bit **checksum** field which like UDP uses the standard Internet checksum algorithm to protect the whole segment, plus the IP pseudo-header.

The final field in the standard TCP header is the 16-bit **urgent pointer** which we have already mentioned and which points to the byte position in the segment where the last byte of urgent data can be found.

The option field, which is often empty, is used to provide some extra facilities, the most important of which allows TCP implementations to indicate the **Maximum Segment Size (MSS)** that they are prepared to accept on a connection. The MSS includes the transport layer headers but excludes network layer headers and data link layer headers. It is set to a value to avoid fragmentation of IP datagrams at the network layer. Routers can normally handle datagrams up to 576 bytes long without fragmentation. The MSS is therefore typically set to 536 for wide area networking and 1460 for local area networking, allowing a standard 40 bytes for the IP header and 1500 bytes for the data field in an Ethernet frame.

5.5.2 Connection control

As a connection-oriented protocol, TCP has a discrete number of connection states, such as ESTABLISHED and CLOSED and rules about events that cause a change of states. The protocol can therefore be modelled and implemented as a **Finite State Machine.**[6]

[6] *Tanenbaum, Chapter 6.3.3, page 522 and Chapter 6.5.7, page 541.*

TCP has a connection control function involving three-way handshakes that ensure connections are opened reliably with both ends of the connection in agreement on the connection state, even when segments are lost, duplicated or delayed.

Although TCP is designed as a peer-to-peer protocol, where either end can initiate a connection, it is often used in a client server mode where the client always initiates a connection, as it does with HTTP. When describing the operation of the protocol, it is often convenient to assume a client server relationship and then refer to the connection initiator as the client and the connection acceptor as the server.

To establish a connection, the client initiates a connection request to the server. This request is a TCP segment without any data and with the SYN flag set. In order to accept a connection, the server responds by sending a TCP segment without any data and with both the SYN and ACK flags set and with the acknowledgement number incremented by one. When this is received the client responds with an acknowledgement by sending another segment with the SYN flag set and with the acknowledgement number incremented by one.

Established connections need a unique connection identifier. This is formed by joining the client IP address, the client port number, the server IP address and the server port number. Although the server IP address and port number could be used by many simultaneous connections, the uniqueness of the connection identifier is guaranteed by the dynamic allocation of unique combinations of IP addresses and port numbers for the clients.

The three-way handshake is also required to close a connection, to cope with the possible loss of the segments that are needed to close the connection, but there is an additional problem that a reliable transport protocol like TCP must overcome. It must ensure that all outstanding data segments are successfully received before closing the connection. This problem does not have an infallible solution. It is akin to a classical problem called the two army problem[7] which can be shown to have no complete solution. TCP minimises the risk of losing data segments during the closure process, by operating timers at both ends, and if they do not have confirmation of closure after these timers have expired, they will unilaterally close the connection.

[7] *Tanenbaum, Chapter 6.3.3, page 502.*

Activity 5.4

Run the **netstat** or **netstat | more** command from a DOS/Unix prompt. Piping the output of netstat through more will definitely be required, if you execute the command on a Unix server. Note the TCP connections that are listed and their current states. Now run some new applications, like a web browser, and mail, FTP or Telnet clients, and then run netstat again. Find the new connections that these applications have established and note their states.

5.5.3 Segmentation

Once a socket has been successfully opened, the application process will write to the socket in a similar way that it writes to a file. The socket will receive a stream of characters. TCP must segment this data prior to transmission. It must take account of the receiver's Maximum Segment Size if it was indicated when the connection was established or make use of the default MSS (usually 536 bytes). At the receiving end, TCP must remove all trace of the segments and deliver a character stream for its application process to read.

5.5.4 Error control

TCP does not have any negative acknowledgements. Errors must be indicated by the absence of positive acknowledgements. A segment is acknowledged by setting the acknowledgement bit in a segment being sent back to the transmitter and indicating in the acknowledgement field the next sequence number (representing the next character in the data stream) that the receiver is expecting.

Acknowledgements can be piggy-backed on data segments that are being transmitted, or they can be sent in segments that contain no data.

TCP checks all characters in the segment using the Internet checksum and if there is an error, it will return an acknowledgement with the acknowledgement number indicating the first byte in the segment that failed the error check. This will cause the transmitter to retransmit the segment.

The transmitter also sets a timer whenever it transmits a segment. If no acknowledgement for a segment has been received when the timer expires, the segment is retransmitted.

The choice of value for this retransmission timeout will affect the efficiency of the protocol. If it is too short, there will be unnecessary retransmissions, but if it is too long, TCP will be slow to react to segment loss. TCP actually dynamically adjusts this timeout value by monitoring the Round Trip Times (RTTs) between segments being sent and acknowledged and calculating the appropriate retransmission timeout. A number of different schemes for doing this have been proposed at different times. The timeout is usually a function of a weighted moving average of RTTs and their standard deviation. Each time a segment is retransmitted, TCP will double the retransmission timeout and when the acknowledgement is received, TCP will return to calculating the retransmission timeout from the weighted moving average and standard deviation.

TCP does not specify what the receiver should do with any segments received after a segment that has failed its checksum. Some implementations will discard all subsequent segments and expect the transmitter to retransmit them all. Other implementations will buffer the subsequent segments, and on receipt of the retransmitted segment will then acknowledge all the subsequent segments, by indicating the next byte it is expecting. This is known as a **cumulative acknowledgement**. The transmitter, provided the retransmission timers for the subsequent segments have not expired, will not then have to retransmit all the segments that were successfully received.

5.5.5 Flow control

TCP uses a sliding window mechanism for flow control, that makes use of the window sequence number and acknowledgement number fields. Each TCP implementation has a receive buffer where it stores incoming data before it is read by the application process. TCP must keep track of the last byte received from the transmitter and the last byte read by the application and calculate the number of unread bytes in the buffer. It then calculates its window from the difference between the size of the receive buffer and the number of bytes in the buffer that have not been read. It advertises the current value of its window to the transmitter every time a segment is sent back to the transmitter.

The transmitter must also keep track of the last byte that it sent and the last byte that the receiver has acknowledged. It subtracts them to calculate the number of unacknowledged bytes that it has sent. It then has to make sure that the number of unacknowledged bytes never exceeds the last value of the receiver's window that was received.

By this mechanism, the receiver can control the rate at which the transmitter sends data, so that the receive buffer never overflows.

5.5.6 Congestion control

Flow control applies to individual TCP connections, but congestion will affect a large number of connections that are routed through a part of the network that is congested. A mechanism is also required to control the flow of data on all the connections affected by the congestion in order to relieve the congestion. The TCP congestion control mechanism has been designed to do this. The congestion control mechanism will cause the transmitters on all the TCP connections routed through a congested part of the network to be slowed down.

It does this by using another window called the **congestion window** alongside the receive window. The transmitter then has to ensure that the number of unacknowledged bytes it has sent does not exceed either the receive window or the congestion window.

Like the receive window, the congestion window is adjusted dynamically by means of an algorithm. The congestion window is intialised to 1 MSS when the connection is established. In the first stage, it grows exponentially from a slow start, doubling every time a segment is sent until it reaches a threshold (initially set at 64 Kbytes). Once the threshold is reached, the congestion window then grows linearly. At any point, if a retransmission is required, there is a sudden drop of the congestion window back to 1 MSS and at the same time the threshold is halved. This part of the algorithm is therefore described as additive increases, multiplicative decreases. Following the reinitialisation of the congestion window, it then grows exponentially again until the new threshold is reached or a retransmission occurs.

The overall effect of the congestion control mechanism is that the throughput on TCP connections is not constant. The throughput will follow a saw tooth pattern where it grows exponentially from a slow start, then linearly and then plunges back to its starting value and then grows exponentially again. This is one of the main reasons why TCP is totally unsuited to transport real-time traffic such as voice and video.

The good thing about the algorithm though is that it is excellent at relieving congestion. Once congestion occurs, all TCP connections affected dramatically reduce their throughput, which very quickly relieves the congestion. In fact, if it wasn't for the TCP congestion control algorithm, the Internet would probably not function well at all. The majority of Internet applications use TCP and these can respond quickly to relieve congestion. If all applications used UDP, there would be no way to relieve congestion. Datagrams would just have to be discarded and this would result in retransmissions which would worsen the congestion, to the point that probably very little traffic would get through.

At this point, it would be useful to summarise the main differences between the TCP and UDP protocols.

Table 5.3: Comparison of TCP and UDP protocols

TCP	UDP
Connection-oriented	Connectionless
Provides a reliable service	Provides an unreliable, best efforts service
Complex protocol	Simple protocol
Stateful protocol	Stateless protocol
Large protocol overhead	Small protocol overhead
Stream-oriented	Message-oriented
Good congestion control mechanism	Can do nothing to alleviate congestion
Well suited to data applications but not real-time (video and voice) applications	Well-suited to real-time (video and voice applications), but not data applications, other than short client server transactions
Only really suited to point-to-point applications	Well-suited to multicast applications

5.6 Sequenced Packet Exchange (SPX)

SPX is a connection-oriented reliable transport protocol developed by Novell for use on NetWare LANs. It provide similar functions to TCP, but is not as complex. Although many NetWare installations are now configured to use TCP/IP, there are still some SPX implementations about. SPX is a virtual circuit protocol. SPX packet headers contain source and destination connection IDs which are equivalent to port numbers. Like TCP, the header also has a sequence number, acknowledgement number and the equivalent of a receive window.

5.7 Transport Layer Security (TLS)[8]

[8] *RFC 3546*

TLS sits between application layer protocols and TCP. It could therefore be regarded as belonging to either layer. Conventionally, it is usually described as a transport layer protocol. It is sometimes also called Secure Sockets Layer

(SSL) from a protocol originally developed by Netscape for their Navigator browser to access secure web applications. It can also be used with other applications. It is a simple protocol that provides two main functions. It is used to authenticate the identity of the server to the client (and optionally the client to the server) and it also provides end-to-end encryption to protect the data being transmitted.

Before we look at TLS, we need to learn something about private and public key encryption and digital certificates.

Symmetric key encryption, also sometimes called private or secret key encryption, uses the same secret key for encrypting a message as for decrypting it. It uses a mathematical function that takes the message and the key as its input and produces an encrypted message as its output.[9] With this method, both parties to the communication must have knowledge of the secret key and to be secure, nobody else can have knowledge of it.

[9] *The Data Encryption Standard (DES) is an example of a symmetric key encryption algorithm that uses a 56-bit key.*

Public key encryption[10] uses an algorithm which is based on a one-way function that takes the message and a public key as its input to produce the encrypted message by a relatively easy calculation. But the function is very hard to reverse. If you have an encrypted message and the public key, it is very hard to calculate what the original message was. Public key encryption uses a public key to encrypt a message and another private key to decrypt it. Organisations can therefore publish their public keys, but will keep their private keys secret.

[10] *RSA, named after its inventors (Rivest, Shamir and Adleman) is an example of a public key encryption algorithm.*

Secure authenticated communications can use public key cryptography to authenticate servers and clients. A message is encrypted by a client using its own private key and then it is encrypted again using the server's public key. At the server the message can be decrypted using the server's private key and then decrypted again using the client's public key. This process authenticates both parties and also ensures privacy, integrity and non-repudiation. The only problem is how to obtain the public key for any organisation with which the client wishes to communicate. The answer to this problem is to use a Public Key Infrastructure (PKI) and Digital Certificates.

Digital Certificates are issued by trusted third parties, called Certification Authorities (CAs), whose public keys are well-known, to verify the authenticity of organisations issuing public keys. The organisation's public key is encrypted using the CA's private key. The server then can forward this digital certificate to a client, who can decrypt it using the CA's public key to obtain the server's public key. The client, because it trusts the CA, can be assured that the server and its public key are authentic.

TLS consists of two protocols:

- **TLS Handshake Protocol** that negotiates security algorithms, authenticates the server to the client and optionally defines other communications parameters. The client sends a hello message to the server which indicates the version of TLS being used. The server sends a certificate message which contains a digital certificate encrypting its own private key with the CA's private key. The client then generates its own secret key and sends it to the server encrypted with the server's public key. The client then sends a message encrypted with its secret key, to terminate the handshake. The server decrypts this message and responds with a message encrypted with the secret key that terminates the handshake.
- **TLS Record Protocol** that encapsulates the TLS Handshake Protocol and carries all subsequent TCP segments encrypted with the secret key using the algorithm negotiated by the handshake protocol.

Activity 5.5

Using a textbook or web searches, read more about the DES and RSA algorithms and digital signatures and certificates.

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
   i. The transport layer is the lowest layer that operates end-to-end between two application processes.
   ii. The TCP sequence number is always initialised to 0 at both ends of a TCP connection.
   iii. A TCP acknowledgement contains the sequence number of the last segment that was successfully received.
   iv. The TLS Handshake Protocol negotiates which algorithm and which secret key will be used for encryption.

b. Describe three possible ways that a transport layer can be implemented. Which method is used in Berkeley Unix and in Winsock?

c. Calculate the Internet checksum for 16-bit words: 10101100 10010101 and 00100100 11100100. Show how the checksum will detect an error that inverts the third and fourth bits from the right of the first word.

d. List the names of the four fields in the UDP header and state their lengths in bits.

e. Describe how the TCP congestion control mechanism works. Explain what effect this might have on real-time applications.

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 transport layer
- describe the functions of the transport layer
- calculate and check an Internet checksum
- describe the header field and operation of UDP
- describe the segmentation and connection, error, flow and congestion control functions of TCP
- outline the differences between UDP and TCP
- outline how Transport Layer Security works.

Chapter 6: The network layer

Introduction

In this chapter, we shall examine the services and interfaces provided by the network layer and the general functions of network layer protocols.

We shall concentrate our attention specifically on the Internet Protocol Versions 4 (IPv4) and other network layer protocols used in the Internet. The other network layer protocols we shall consider are the Internet Control Message Protocol (ICMP), the Internet Group Management Protocol (IGMP), the IP Security Protocols (IPsec) and the Address Resolution Protocols (ARP and RARP). We shall also consider the deficiencies of IPv4 and how they are addressed in the next generation Internet Protocol Version 6 (IPv6).

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, (2003). Chapters 20, 31.1

Kurose and Ross *Computer Networking – A Top Down Approach featuring the Internet* (Addison Wesley), third edition (2005). Chapters 4.4.1, 4.4.2, 4.4.3, 4.4.4, 4.4.5, 4.7, 4.8.2, 7.8.3.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition (2003). Chapters 18, 21.6.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002). Chapters 5.1, 5.6.1, 5.6.2, 5.6.3, 5.6.6, 5.6.8, 8.6.1.

6.1 Services

The network layer is concerned with moving data across one or more subnetworks between hosts. The path through the subnetworks will normally involve a series of network layer devices, known as routers or switches, interconnected by data links. The network layer hides all the details of these subnetworks from the transport layer. The network layer is not concerned with delivering data to application processes, it simply has to deliver data to the correct host.

As with the transport layer, there are two distinct network services that can be offered. A connectionless network service[1] (often in the network layer referred to as a **datagram service**) and a connection-oriented network service[2] (often in the network layer referred to as a **virtual-circuit service**).

[1] The Internet Protocol is an example of a connectionless (or datagram based) network service.

[2] The X.25 Packet Layer Protocol and the Asynchronous Transfer Mode Protocol are examples of connection-oreinted (virtual-circuit based) network layer protocols.

6.1.1 Connectionless network services

With the connectionless network service, each datagram is completely self-contained with a full destination network layer address in its header and each packet is routed independently. The path the datagram takes through the subnetworks is determined dynamically by routers reading its destination address and taking a routing decision about where to forward the datagram. Consequently, a sequence of datagrams between two hosts could follow completely different paths and arrive in an order different to that in which they were transmitted. Connectionless protocols also cannot reserve any memory resources in the routers or transmission capacity on the data links, as it is impossible to determine in advance which routers will actually forward the datagram and over which data links. Because resources cannot be reserved for transmitting datagrams, quality of service in connectionless network layers can be very variable, as reliability and performance will depend on how much other traffic is being routed. Service will be degraded if the network becomes congested. Connectionless network services therefore offer no guarantees regarding the delivery of data in order and without loss. They provide a best effort service. Because no resources are reserved and each datagram is self-contained and routed independently, routers do not hold any state information. If a router crashes and restarts, all the datagrams it had in buffer will be lost, but after it has been restarted any new datagrams received can be forwarded, as before. Connectionless network layers are simple and flexible.

6.1.2 Connection-oriented network services

Connection-oriented network services, on the other hand, must set up a connection through a series of network layer devices, called switches, before they can send data and this connection must be closed down after all the data has been transferred. Note that a network layer connection is very different from a transport layer connection in that the former involves intermediate devices and the latter only the two hosts. The connection can be circuit-switched using dedicated transmission capacity or packet-switched where transmission capacity is shared dynamically. Connection-oriented packet-switched network layers set up **virtual circuits** through network devices, while circuit-switched networks, such as the PSTN, set up real dedicated circuits. In this course, we will mainly concern ourselves with packet switching, as today virtually all digital data and increasingly also voice and multimedia communications are packet-switched. Connection-oriented packet-switched networks are often called **virtual circuit networks**.

With connection-oriented networks, all packets between a source and destination sent over a virtual circuit will follow the same route. The connection initiator can indicate how much resource is required, such as transmission capacity on data links and memory buffers in switches. This allows the network to be able to guarantee quality of service. Note that packets in a connection-oriented network layer are not called datagrams, as they do not contain a destination network address in their headers. Instead, they contain a shorter **virtual circuit identifier**. The virtual circuit identifier is specific to each data link. The switch examines the virtual circuit identifier on incoming packets and forwards the packet to another switch using a different virtual circuit identifier. Each switch has to maintain a mapping of incoming virtual circuit identifiers to outgoing virtual circuit identifiers for each virtual circuit. This is an example of state information. If the switch crashes, this information will be lost and the switch will then not know how to forward packets. The host that set up the virtual circuit will have to determine that the connection has been broken and will have to re-establish it.

Connection-oriented network layers are much more complex than connectionless network layers. With connectionless network layers, most of the intelligence and complexity is in the hosts, but with connection-oriented network layers, a large part of the intelligence and complexity resides in the switches. Many network operators prefer to offer connection-oriented network services, as it means that they are much better able to control the quality of service, especially when the network becomes congested. As a result of being able to guarantee quality of service and adding value by all the work that is done by the switches, they can justifiably charge more for the network service they provide.

6.2 Interfaces

The interface offered by the network service will depend on whether the service is connectionless or connection-oriented. The most basic connectionless service will have primitives for sending a datagram and receiving a datagram. A connection-oriented network service will have additional primitives for establishing, resetting and closing virtual circuits.

The network service primitives will be called using parameters to define such things as destination address and to specify other protocol header fields and options. Application developers will not usually directly use a network service primitive. They will use the transport service primitives.

6.3 Functions

The network layer has two main distinctive functions. Firstly, it is responsible for path determination which involves maintaining and distributing routing tables by means of routing protocols, so that routers know where to forward packets. This function is also required in virtual circuit networks and circuit-switched networks, in order that the switches can determine how to set up a connection.

Secondly, for packet switched networks, the network layer is responsible for packet switching which is the process of actually forwarding the packets, according to the routing tables or by mapping virtual circuit numbers. In this subject guide, we will concentrate on packet switching and defer the study of routing to the second half of the course.

In addition to these functions which are specific to the network layer there are also some generic functions which the network layer shares with other layers.

6.3.1 Encapsulation

The network layer encapsulates transport layer SDUs with a network layer header to form a network layer PDU. At the receiving end, the network layer will process the network layer header, decapsulate the transport layer SDU and pass this to the transport layer.

6.3.2 Addressing

Network layer addresses must be globally unique, so that no two network layer devices can have the same address. Destination addresses are used to route packets or to set up virtual or real circuits. An address that has been duplicated will cause confusion and the misrouting of data. The allocation of addresses therefore has to be managed carefully.

Network layer protocols tend to use logical addresses, configured in software, as opposed to physical addresses, which are burned into LAN Network Interface Cards (NICs) and are used for addressing by LAN data link layer protocols. This is done to keep the network layer independent from the lower layers. It means that a faulty NIC card on a server can be replaced without affecting the network layer. A process, called **address resolution**, is therefore needed to map logical network addresses to physical addresses.

Some network layers, such as Novell's Internet Packet Exchange (IPX), designed for use with Novell's Netware Network Operating System, do not maintain this independence and embed the 6-byte physical address within a

12-byte network address. Such network layers do not need to perform address resolution. Netware also supports TCP/IP and many installations are now migrating from IPX to IP.

6.3.3 Multiplexing

The network layer is responsible for transmitting data between hosts, but not between application processes, which is a function of the transport protocols. The receiving network layer will receive a sequence of packets and it is possible that some of them could encapsulate different transport protocols or even another (higher) network layer protocol. The receiving host must know how to handle these packets and to pass them to the correct process, so a field in the network layer header is required to indicate the protocol being carried. The receiving network layer can then de-multiplex the received packets and forward them to the correct process.

6.3.4 Segmentation

One of the issues that many network layers must address is how to deliver large packets over data links that have a limited frame size. To be able to do this, network layer protocols must be able to segment the data into smaller units so that they can each fit into the maximum size frame supported by the data link layer called the **Maximum Transfer Unit** (**MTU**). The network layer protocol must have an identification field, so that the receiver can identify all the segments created from the original packet. In the case of a connectionless network layer, the segments must also contain a sequence number, so that the receiver can reorder the segments if necessary before reassembly.

6.3.5 Ordered delivery

Where the protocol is connection-oriented, the network layer will only pass them on to the layer above in the correct order.

6.3.6 Flow control

Connectionless network layers will typically not have a flow control mechanism. If data is being received by a host or a router at a faster rate than it can be handled, the network layer will just discard those packets it cannot handle and rely on upper layers to detect the loss and retransmit the packets.

Connection-oriented network layers will have some form of flow control mechanism, so that they can slow down the speed at which data is being transmitted. They normally do this by means of acknowledgements and retransmissions, often using a sliding window mechanism, so that a number (the window size) of unacknowledged packets can be transmitted before the flow of data is controlled.

Connection-oriented networks also have other means of flow control when network congestion is occurring. They can route new connections away from any part of the network that has become congested or they can in extreme circumstances refuse to accept any new connections, until the congestion has been alleviated. This latter mechanism is know as **admission control**.

Another means of flow control is to issue a source quench or choke packet to temporarily slow down a transmitter. Although IP has no flow control mechanisms, IP implementations can make use of source quench packets by means of another protocol, that is implemented with IP, known as the Internet Control Message Protocol (ICMP).

6.3.7 Error control

Similar to flow control, error control is likely to be required in connection-oriented protocols and not required in connectionless protocols apart from simple error detection and discarding. Where required, it will make use of sequence numbers, checksums and acknowledgements to ensure that data is delivered without errors.

6.3.8 Grade of service

The main cause of network congestion is the unpredictable and bursty nature of traffic, which can fluctuate between being heavy and light over short periods of time. In order to avoid congestion and to be able to guarantee quality of service a network layer must be able to smooth out the traffic that it is offered. This process is known as **traffic shaping**. When a virtual circuit is set up, the transmitter attempts to predict its traffic pattern by specifying certain parameters in a **flow specification**. The network layer in accepting the connection agrees to support this pattern and guarantees that capacity will be available to meet the requirements. The sender can ensure that it conforms to the agreed traffic pattern by instituting mechanisms such as the leaky bucket[3] or token bucket algorithms[4]. The network operator must also carry out **traffic policing**, to ensure that the flow specification is not being exceeded.

[3] Tanenbaum, Chapter 5.4.1, page 400.

[4] Tanenbaum, Chapter 5.4.1, page 402.

Only connection-oriented networks can support grade of service, as in order to be able to deliver it, resources such as buffer space in network nodes and transmission capacity on links need to be reserved and connectionless networks are unable to do this. Asynchronous Transfer Mode (ATM)[5] is a protocol suite which has been designed to support different grades of service.

[5] The abbreviation ATM should not be confused with Automatic Teller Machines. These are the machines used by banks to dispense cash to customers and which are linked to the banks mainframe computers and other banks' computers via data networks, often also confusingly referred to as ATM networks.

Activity 6.1

From a texbook or a web search, read about the leaky bucket and token bucket algorithms.

6.3.9 Security

There is much debate about whether security should be provided in the network layer, or in higher layers. Some network layers do implement security functions. IPsec protocols have been developed to provide security functions for the IP network layer. These have now been incorporated into IPv6.

6.4 Internet Protocol version 4 (IPv4)[6]

[6] RFC 791.

IPv4 has become the network layer of choice for most data communications. IPv4 is a connectionless protocol that can carry datagrams of up to 64 Kbytes. Strictly speaking, it is an internetworking protocol, which means that it is designed to enable data to be carried across a series of many different types of subnetwork (LANs, MANs and WANs).

Because IPv4 is connectionless, it only requires two primitives: SEND to transmit a datagram and DELIVER to notify the arrival of a datagram. Both primitives indicate the sources and destination IPv4 addresses, protocol, type of service, data length, options and the data itself. The SEND primitive can also indicate the Time to Live and the Don't Fragment indicator.

It should be noted that no application developer is likely to make direct use of these primitives. Instead they would use the socket TCP or UDP interfaces provided by operating systems. The IP primitives would only be used by transport protocol developers, who are few in number.

6.4 Encapsulation

IPv4 encapsulates data received from the higher layer by adding an IPv4 header which will introduce 20 or more bytes (depending on options) of protocol overhead. The IPv4 header structure was studied in a Level 1 unit and is defined in virtually all text books on data communications. You should make sure that you understand the fields of the header and how they are used.[7]

[7] *Tanenbaum, Chapter 5.6.1, page 433.*

6.4.2 Addressing

IPv4 addresses being four bytes long, can theoretically support 2^{32} or just over four billion addresses. However the address space is structured and some of the address space is wasted or reserved. The structure and representation of IPv4 addresses and address classes was studied in a Level 1 unit.

Table 6.1 shows the Classful IPv4 address scheme which is now obsolete. It is called classful as IPv4 addresses belong to one of five classes[8] and to distinguish it from the classless addressing scheme that replaced it. Note that there are some network and host ids that have special meanings and cannot be allocated as addresses.[9]

[8] *An IP address beginning with 127 does not belong to a class. These addresses are reserved for carrying out loopback tests and any datagrams with these addresses will be passed straight from the transmit side to the receiver side of an IP interface.*

[9] *Network identifiers of all 0s means the local network and network identifiers of all 1s means all networks. Host identifiers of all 0s means the network itself and host identifiers of all 1s is the broadcast address for every host on the network.*

Table: 6.1: IPv4 Classful Addressing Scheme

Class	Class ID (first bits)	1st Byte	Net Id (bytes)	Number of Networks	Host ID (bytes)	Number of Hosts	Use
A	0	1-126	1	126	3	16,777,214	Large Networks
B	10	128-191	2	16,382	2	65,534	Medium Networks
C	110	192-223	3	2,097,150	1	254	Small Networks
D	1110	224-239					Multicast
E	1111	240-255					Experimental

The main reason for splitting network addresses into two parts is that routers do not need to have any entries about host addresses in their routing tables. They can determine the class of the network address from the first few bits and then look up the network identifier in a routing table to find where the datagram should be routed next.

The above addressing scheme led to a number of problems, particularly with Class B networks where demand for network addresses was much too high. Some respite was given by borrowing some bits from the host part of the address to create addresses for subnetworks. An organisation could therefore be allocated just one Class B address and use a few bits from the host identifier to define a number of subnetworks. This process is known as **subnetting.**[10] IP requires that all physical media such as LANs are allocated

[10] *RFC 950*

a separate subnetwork address. The same is even true of point-to-point circuits with only two devices attached to them. Without subnetting, address space would very soon be exhausted. Routing tables in the Internet only need to have a single entry for the classful network identifier. Routing tables within the organisation's network would have entries for each of the subnetworks and routers can apply a subnet mask to the destination address to determine the network identifier (including the subnetwork identifier) which can then be looked up in the routing table. Subnetting and subnet masks were studied in a Level 1 unit. You should make sure that you understand how subnetting works.

Classless Inter-Domain Routing (CIDR)

Further problems with the demand for Class B addresses resulted in another initiative that led to the abandonment of classful addresses. This initiative is known as **Classless Inter-Domain Routing** (**CIDR**).[11] It is sometimes also called **supernetting**. The main problem with classful addressing was that most organisations, even though they may have had less than 253 hosts, often felt that they might need to grow beyond that and hence applied for Class B addresses rather than Class C addresses. But these addresses were being used very inefficiently as only a small number of the 65,534 host addresses were being used. Furthermore, Internet Service Providers, who had not been allocated Class A addresses, were applying for many blocks of Class B addresses for their customers. This resulted in a huge growth in the size of routing tables, as each ISP customer network had an entry in the table even though the routing required for each of the networks was the same, as they all had to be forwarded to the same ISP. The answer to both of these problems was to implement CIDR and abandon classful addresses. Under CIDR, network addresses can be of any length and are not forced to be a multiple of 8 bits. As routers are no longer able to determine the length of the network identifier from the class of the address, information about the length of the network identifier has to be included in the routing table, so that the correct subnet mask can be applied when checking destination addresses against entries in the routing table. The length of the network identifier in bits is shown at the end of the IPv4 address after an oblique stroke. It is known as the **prefix length**. This notation can also be used to define subnetworks, in which case the subnet mask consists of a number of 1s equal to the prefix length followed by 0s.

[11] *RFC 1519.*

The process of looking up an address in the routing table is therefore: for each entry in the table, apply the appropriate subnet mask to the destination address and attempt to match it against the network identifier in the table and then choose the longest network identifier that matches the masked destination address.

For example, if a network is defined in a routing table as 192.10.64.0/19, its network identifier is 19 bits long.

In binary (with spaces showing the byte boundaries) the address will be:

11000000 00001010 01000000 00000000 = 192.10.64.0

If we split this address with a "|" to show 19 bits in the network id we get:

11000000 00001010 010|00000 00000000 = 192.10.64.0/19

So the first host address in this range will be:

11000000 00001010 010|00000 00000001 = 192.10.64.1

and the broadcast address will be:

11000000 00001010 010|11111 11111111 = 192.10.95.255

and the last host address will be:

11000000 00001010 010|11111 11111110 = 192.10.95.254

In all, there will be (95 – 64 + 1) * 256 = 8,192 addresses in this range,

and (95 – 64 + 1) * 256 – 2 = 8,190 host addresses in this range.

The net mask will be:

11111111 11111111 111|00000 00000000 = 255.255.224.0

Now suppose we need to look up a datagram with the destination address 192.10.66.2 which will be in the range of this network.

We firstly, bitwise AND the address with the net mask:

11000000 00001010 010|00010 00000010 = 192.10.66.2

11111111 11111111 111|00000 00000000 = 255.255.224.0

11000000 00001010 010|00000 00000000 = 192.10.64.0

Which can then be looked up in the routing tables and will match the entry for the network.

Variable Length Subnet Masks (VLSM)

The standard method of defining subnets means that all the subnets in a network have the same number of hosts. This can be inefficient, particularly for point-to-point circuits where a whole subnet address is used for just two hosts. VLSM allows a network to be partitioned into a number of subnets of different lengths, but it can only be implemented where a routing protocol that supports VLSM is used.

Activity 6.2

Check the IP address(es) on your computer and netmask(s) configured on your computer, at home, at your institution or work or at any other location where you have access to the Internet. You can do this on most operating systems by running a program called "ipcfg", "ipconfig" or "ifconfig". Windows PCs will support ipconfig, ipcfg or winipcfg (depending on the version of Windows) and these programs can be run from a DOS command prompt (more information can be obtained by using the /all option after the command). A similar facility exists on Unix based systems and is usually called "ifconfig".

Use the netmask(s) you discover to determine the length of the network id and the host id and then work out whether subnetting or supernetting is being used and the range of addresses that are supported on any subnet or supernet. Look up your IP address(es) in the whois server at the appropriate Regional Internet Registry, which will confirm who owns the address and the range of addresses to which it belongs.

Table 6.2: Regional internet registries

American Registry for Internet Numbers (ARIN)	North America and sub-Saharan Africa	http://www.arin.net/whois
Asia Pacific Network Information Centre (APNIC)	Asia / Pacific	http://www.apnic.net/whois.html
Latin American and Caribbean Internet Address Registry (LACNIC)	Latin America and some Caribbean Islands	http://lacnic.net/en
Réseaux IP Européens Network Coordination Centre (RIPE NCC)	Europe, Middle East, Central Asia and North Africa	http://www.ripe.net/whois

6.4.3 Multiplexing

IPv4 datagrams are received from a higher layer protocol such as TCP to be sent to a destination host. When the datagram is received at the destination, the IPv4 implementation must be able to determine which higher layer protocol software should receive it. The choice is usually between TCP and UDP, but it could also be the Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP) or other transport or network layer protocols. A destination could receive a stream of datagrams from the same source, some of which contain TCP segments, some UDP datagrams and some ICMP datagrams. In effect, IPv4 multiplexes all of these different protocols and the destination IPv4 implementation must demultiplex them and forward them to the correct processes to handle them. IPv4 achieves this by means of the protocol field in the IPv4 header which indicates the protocol the datagram is carrying. Common values of the protocol field are: 1 for ICMP; 2 for IGMP; 5 for TCP; and 17 for UDP.

6.4.4 Segmentation (Fragmentation)

In IPv4, segmentation is called fragmentation, to avoid any confusion with TCP segments.

The maximum size of an IP datagram is 65,535 bytes, so in theory, a datagram of this size could be passed over the IP service interface. Very few, if any, subnetworks could support such a large datagram. Ethernets have an **Maximum Transfer Unit** (**MTU**) size of 1500 bytes and some IP implementations will only support MTU sizes of 536 bytes. In order for IP to be able to handle large datagrams, it must be able to fragment them.

When IPv4 is required to send a datagram that is larger than the MTU of a sub-network, IPv4 will fragment the datagram into a number of fragments. Each fragment, apart from the last fragment, will have the More Fragment bit set in its IPv4 header. The last fragment will not have this bit set. Each fragment will have the same identification number in its header, but the fragments will be identified by means of the fragment offset field, which will indicate the position of the fragment in the original datagram measured in units of 8 bytes starting at 0.

IPv4 only ever reassembles fragments at the destination host. It could in theory reassemble the fragments at the router after the data link with the restricted MTU size, but the datagram might then need to be fragmented again should another data link with a restricted MTU size be encountered. To avoid the possibility of repeated fragmentation and reassembly, the IPv4 designers chose to only reassemble at the destination host.

At the destination host IPv4 stores all the fragments that have the same identification number and will reorder them if necessary, according to their fragment offset fields. If one or more of the fragments has not arrived before a timeout expires, all the fragments are then discarded and the higher layer protocols have to retransmit the data. Once all the fragments have been received and have been reordered the original datagram is reconstructed and passed to the higher layer.

Activity 6.3

Discover the size of the largest datagram that can be transmitted from different places such as your institution, place of work and home. You can do this by using the ping command which is provided with most operating systems. Ping sends out a series of ICMP echo request packets containing data to a specified host who responds with an ICMP Echo Reply packet. The ICMP protocol will be discussed later in this chapter. You can also set the Don't Fragment (DF) bit using the ping command. You may need to check the exact syntax of the command as it varies from system to system. The following command lines work for the DOS version and Unix respectively.

```
DOS:          ping –l 64000 –f hostname
Unix:         ping –c 1 –s 64000 –M do hostname
```

Note: some versions of ping do not allow the DF bit to be set and some versions need to be interrupted with a Ctrl-C to quit.

Both of these versions will respond with error messages, as the datagram size of 64000 will be larger than the MTU size of any subnetwork, but the Unix version will helpfully inform you what the MTU size is. The largest datagram that you are able to send without fragmentation will be less than the MTU size, due to the IP and ICMP headers, but is likely to be between 500 and 1500 bytes. Check to see whether you can send a 500 byte datagram without fragmentation and also a 1500 byte datagram. Then use a binary search technique to find the exact size of the largest datagram you can send, by trying 1000 bytes and then either 750 or 1250 bytes and so on until you can determine the exact size.

6.4.5 Ordered delivery

As a connectionless protocol, where datagrams can follow different routes within the network, IPv4 does not have any function to reorder datagrams, other than the reordering of fragments before reassembly discussed above.

6.4.6 Flow control

IPv4 does not perform any flow control. If an IP implementation is receiving data faster than it can forward it, then the only course of action it can take is to discard datagrams. Higher layer protocols then have to detect the loss and request that the data is retransmitted by their peer higher layer protocol.

A facility does exist within ICMP (which is a protocol that IPv4 implementations can use for reporting errors) to issue a source quench (or choke) packet to slow down the transmitter, but this facility is rarely used, as it adds to congestion and many IPv4 implementations ignore it.

6.4.7 Error control

IPv4 has no error control for data, but does protect its headers using the Internet checksum. On detecting an error, IPv4 simply discards the datagram. The header checksum has to be recomputed at every router as the Time to Live (TTL) field is decremented at each hop. The main purpose of the TTL field is to detect routing loops and prevent them causing problems. Routing loops are caused by errors in routing tables that result in datagrams going round in circles. If nothing is done to deal with this error, more and more datagrams are likely to join the loop, which will cause increasing congestion. The initial value of the TTL set by the source host varies depending on the operating system and the protocols being carried. It is typically in the range 30–255. The TTL is decremented by one at each router until it reaches 0, whereupon the datagram is discarded, thus preventing the routing loop from causing congestion.

Activity 6.4

Use ping to send out an ICMP Echo Request packet with a TTL of 1. You may have to check with a manual how to do this on you computer. This can be done by entering the following commands on DOS and Unix:

```
DOS:          ping –i 1 hostname
Unix:         ping –c 1 –t 1 hostname
```

This will send out an Echo Request to the host specified, but at the first router on the path to the host, the TTL will be decremented to 0 and the router will discard the packet and send back an ICMP Time Exceeded packet to the sender. But it will provide some useful information, such as the IP address and sometimes even the name of the router (found via a reverse DNS look-up from your computer). The DOS version of ping uses ICMP Echo Request packets, but the Unix versions tend to use UDP starting with port 33434 and incrementing the port number by one for each packet sent. Some firewall routers will allow UDP traceroute packets but not ICMP Echo Requests or vice versa.

You can then change the –i or –t parameter to set the TTL to 2. This time the packet will be forwarded by the first router which will decrement the TTL to 1, and then the second router on the path to the host will decrement the TTL to 0, discard the packet and send an ICMP Time Exceeded packet back to the sender. If you carry on with this process, you can obtain all the IP addresses and possibly names of the routers on the path to the host. A program was developed by Van Jacobson to carry out this process automatically, it is called traceroute (or tracert on DOS). It is also a useful diagnostic tool for network managers, as it can pinpoint the source of delays and other network problems.

Activity 6.5

Experiment with traceroute by tracing the routes to various different hosts. Record the names of the first few routers on these routes and attempt to draw a map showing how the these routers are connected and which routers (and which ISPs) carry traffic to which destinations.

Activity 6.6

Visit the site http://www.traceroute.org and select a site in another country. Use it to trace the route back to a web site in your own country. Finally, there are some interesting projects that use traceroute to build up maps of the Internet. The following links are worth visiting: http://www.caida.org/tools/visualization/mapnet/Backbones and http:// www.cybergeography.org/atlas/atlas.html.

6.4.8 Grade of service

The IPv4 header has a Type of Service (TOS) field that was designed to indicate the grade of service required for the datagram that routers should take account of when queuing or discarding packets, but was rarely used. The TOS field has now been redefined to support **Differentiated Services** (**Diffserv**)[12] and this is currently being deployed in the Internet to provide differentiated quality of service. We will look more closely at Diffserv when we consider multimedia communications in the Enterprise Networking part of the syllabus.

[12] *RFC 2475.*

6.5 Internet Control Message Protocol (ICMP)[13]

[13] *RFC 792.*

ICMP is a network layer protocol that resides immediately above IP. In other words, it is carried within IP datagrams with its header following immediately after the IP header. It was designed to allow IP implementations to report errors and to support some functions to help network managers to diagnose IP and other network problems. It is a simple protocol which now has about 30 message types. The most common message types are:

- **Echo Request and Reply** which are sent out by the ping command as a series of ICMP packets to a device which then echoes them back to the sender and allows the Round Trip Time (RTT) to be measured. We also made use of this type of ICMP packet in Activity 6.2. These ICMP packets provide a very useful diagnostic tool for network managers.

- **Timestamp Request and Timestamp Reply** which are similar to Echo Request and Echo Reply, but there is a field for source and destination host to timestamp the messages.
- **Destination Unreachable** which is sent back to the sender of a datagram by a router when it is unable to forward a datagram. This message type also has a code field which can indicate whether it is the destination network, host, port or protocol that is unreachable or whether the destination network or host are unknown. Network Unreachable code implies that there is a network problem and host unreachable implies that the host is down. Port and Protocol Unreachable are returned from the host if the port or protocol fields in the datagram are invalid. Network Unknown implies that there is a routing problem, as the network portion of the address was invalid and Host Unknown implies that the host portion of the destination address was invalid. The code can also indicate that a datagram cannot be transmitted because fragmentation was needed and the DF bit was set. This ICMP packet was used to help discover the maximum size datagram that could be transmitted in Activity 6.3.
- **Source Quench** which is sometimes issued when a datagram is discarded due to congestion was mentioned when we looked at IP flow control. Its purpose is to slow down the rate of transmission at the sender, but it adds to congestion and it is often ignored. In theory, the sender should slow down until it stops receiving source quench packets and then gradually increase the rate of transmission as long as no further source quench packets are received.
- **Parameter Problem** is used to indicate that a datagram has been discarded because of an invalid field in its header.
- **Redirect** is sent by a router to host when it detects that a host is sending datagrams by a non-optimal route. For instance, if there are two routers on a LAN and the first router receives a datagram where the optimal route is via the second router, the first router will forward the datagram to the second router and send an ICMP Redirect to the host to update its routing table so that datagrams for this destination are forwarded to the second router. In general, routers always know the optimal routes as they exchange routing tables, but hosts do not.
- **Time Exceeded** is sent back to the sender when a router decrements the TTL of a datagram to 0 and discards it. We made use of this in Activity 6.4 and it is used by traceroute in Activity 6.5 and 6.6.
- **Address Mask Request and Reply** allows a host to request its subnetwork mask from its local router.

Activity 6.7

Use the ping command to measure the Round Trip Times (RTTs) to a number of hosts in different parts of the world. Find out the geographical location of the hosts and find their distance from your own location using a web-based distance calculator. Plot a spreadsheet chart showing how RTT varies with distance.

Use traceroute to discover the number of router hops between your location and the hosts. Plot a spreadsheet chart showing how RTT varies with hop count.

Use the ping packet size option (–l or –s) to measure the RTTs to one site, as far away as possible. Vary the packet size from 0 to 1400 in steps of 100. Plot a spreadsheet chart showing how RTT varies with packet size.

If you have access to a Unix system, create a background process to ping a site as far away as possible every 10 minutes and to output the results to a file.

(date; nohup ping –i 600 hostname) > outputfile.txt&[14] should work.

After one week, kill the background process and read the output file into a spreadsheet. Plot a chart showing how RTT varies with time of day.

[14] *The date command is used to timestamp the first record in the file, so that subsequent times can be calculated. The nohup command detaches the process from the user's login process so that the ping process is not killed when the user logs out.*

6.6 Internet Group Management Protocol (IGMP)[15]

[15] *RFC 3376.*

IGMP is a network layer protocol designed to support multicasting. Multicasting is a technique that allows datagrams to be sent to a group of hosts in a single send operation. It requires special multicast routers, that are able to define a cascaded star topology between the sender via other multicast routers to all the hosts in the multicast group. The multicast routers when they receive a datagram with a Class D (multicast) address will copy that datagram to the next multicast router in the cascaded star network. A single datagram is therefore copied and forwarded as multiple datagrams through this cascaded star topology. Without multicasting the sender would have to transmit individual datagrams to each host. With multicasting, the sender only has to send one datagram via its local multicast router. Multicasting is a very useful technique when the same information has to be received by a large number of hosts as might happen if a live event was being transmitted. The multicast routers have to keep track on which hosts belong to which multicast group and IGMP allows hosts to join and leave multicast groups.

6.7 IP Security Protocols (IPsec)[16]

[16] *RFC 2401.*

IP was designed with very little thought given to security. IPsec is an architecture that has been designed to overcome the security weaknesses of IP and to provide for privacy, authentication, integrity and non-repudiation of messages within the network layer. IPsec does not prescribe specific encryption or authentication techniques, but provides a framework in which these can be agreed and can operate.

Despite IP being a connectionless protocol, IPsec is connection-oriented. A security association must be set up between the two parties. In setting up the security association, both parties agree the security algorithms that are going to be used.

There are two modes in which IPsec can operate. The first is **transport mode**, where the IPsec header and encrypted message are encapsulated in an IP datagram in the usual way with the IPsec header following the IP header. The second is **tunnel mode**. In this mode the entire IP datagram including all the headers is encrypted and encapsulated within a new datagram. This is usually done between two secure firewall routers whose IP addresses are used in the new datagram which is then routed through the Internet to the destination firewall router. It is as if the original datagram disappears into a tunnel at the first firewall router and reappears at the second. If this datagram is intercepted on the Internet, its contents including the real source and destination addresses cannot be seen, as they are encrypted. At the destination firewall router the datagram is decapsulated and then decrypted and then the original datagram is forwarded to its true destination.

Two IPsec protocols have been defined. They are Authentication Header (AH)[17] and Encapsulating Security Payload (ESP)[18]. Both AH and ESP can be used in transport mode or tunnel mode. AH authenticates the source host and ensures the integrity and non-repudiation of messages (but not their privacy) by including a message digest using a digital signature in the Authentication Header. ESP will authenticate the ESP header and the data

[17] *RFC 2402.*

[18] *RFC 2406.*

and ensure the privacy, integrity and non-repudiation of the data. In transport mode, it will do this for the IP data field, while in tunnel mode, it will do this for the whole IP datagram.

6.8 Address Resolution Protocols (ARP[19] and RARP[20])

[19] RFC 826.

[20] RFC 903.

Address Resolution Protocol (**ARP**) is a network layer protocol used on a LAN to map an IP address to a physical address. When a router receives a datagram it looks up the destination IP address in its routing table to decide on which interface the datagram should be forwarded. If the interface is a LAN interface, it must forward the datagram over the appropriate LAN protocol, but it will not know what physical address should be placed in the LAN protocol header so that the LAN can deliver it to the correct device. The router resolves this problem by sending out an ARP Request packet containing the destination IP address that it is trying to find and then encapsulates this packet in a LAN frame using the broadcast physical address (all 1s). All hosts on the LAN will receive this frame and will check whether the IP address is the same as the one in their own configuration. If it is not, the host will ignore the ARP packet. If it does have the correct IP address, the host will reply to the ARP Request with an ARP Reply that will be encapsulated in a frame which will have its physical address as the source address. When the router receives the ARP Reply it then knows the physical address to which it must send the original datagram. It caches the mapping between the IP address and the physical address (usually for two minutes) in an ARP table for future use if more datagrams arrive for that destination IP address.

Activity 6.7

Examine the ARP table on your computer by using the following commands:

DOS:	arp -a
Unix:	arp

Reverse Address Resolution Protocol (**RARP**) is a network layer protocol which uses the same formats as ARP, but is used in reverse. With RARP, a device issues a request to discover its own IP address from an RARP server. It can be used by diskless workstations which are unable to store their IP address once switched off. RARP is used less these days, as there are other protocols such as Boot Protocol (BOOTP)[21] (which is used to bootstrap software into diskless workstations, but can also provide IP configuration) and Dynamic Host Configuration Protocol (DHCP)[22] that can also provide IP configurations. Both these protocols run above UDP.

[21] RFC 951, 1048, 1084.

[22] RFC 2131, 2132.

6.9 Internet Protocol Version 6 (IPv6)[23]

[23] RFC 2460–66.

IP Version 4 has been a very successful protocol, but it was designed a long time ago and it has a number of recognised deficiencies, some of which we have already seen:

- it is running out of addresses, because the address is limited to 16 bytes and because of the inefficient structuring of the address space
- routing tables are becoming very large, which slows routing table look-up and causes routing protocols to consume more resources and bandwidth
- its header structure is complex which is difficult for routers to process
- it has well known security weaknesses, as it does not provide authentication and privacy without implementing another protocol (IPsec)
- it does not easily support mobility which is becoming increasingly

important with the growth in use of laptop and handheld computers and wireless data networking.

Internet Protocol version 6 (IPv6) addresses all these problems and more.

6.9.1 Encapsulation

IPv6 encapsulates data received from higher layers by adding an IPv6 header. The IPv6 header has a fixed size of only nine fields and using 40 bytes:

- A 4-bit Version field that defines the version, which will obviously be set to 6.
- An 8-bit Traffic Class field, which is in the same format as the Diffserv Type of Service Field discussed under Grade of Service in IPv4. It also implements Explicit Congestion Notification in the reserved two bits of this field.
- A 20-bit Flow Label field that is used to uniquely identify that a datagram belongs to a specific flow between a source and destination that requires special treatment by routers. This field will be used to identify traffic with critical requirements, such as real-time video.
- A 16-bit Payload Length field performs a similar function to the Total Length field in IPv4, except that the 40 byte IPv6 header is not counted.
- An 8-bit Next Header field defines the type of the next header which may be a header for a higher layer protocol or may be an IPv6 extension header for options.
- An 8-bit Hop Limit field which is just the same as the IPv4 Time To Live field, but with a sensible name.
- Two 128-bit fields for the Source and Destination Address.

The IPv6 header can be followed by any number of IPv6 extension headers for IPv6 options after which is the data field.

6.9.2 Addressing

IPv6 uses 16 byte addresses which creates a theoretical maximum of $3.4*10^{38}$ addresses. This should be more than adequate for any future needs. Addresses can also be structured geographically or by network provider which will then allow either geographical or provider based routing.

IPv6 addresses will be written in hexadecimal format with each group of 16 bits separated by a colon. Leading zeros in each group of 16 bits can be omitted and multiple groups of 16 bits of zeros can be replaced by two colons, but only once in an address.

For example:

1234:0000:0000:0000:0001:2345:6789:ABCD

can be rewritten as 1234::1:2345:6789:ABCD

IPv6 addressing can also accommodate IPv4 addresses in the last 4 bytes. These can still be represented in dotted decimal form for convenience.

For example:

::192.10.1.4 can be used to represent the IPv6 address ::C00A:104 or 0000:0000:0000:0000:C00A:0104 in full

IPv6 addressing will not only support unicast, multicast and broadcast addresses, but will also support a new type of address called anycast. Datagrams will be delivered to just one of a group of addresses, choosing the nearest one as measured by the routing protocol.

6.9.3 Multiplexing

IPv6 has a multiplexing function similar to IPv4, but it does it using a Next Header field in a clever way that allows the IPv6 header to be uncluttered and also for new options to be included in the protocol without changing this header. The Next Header field in the IPv6 Header defines the type of the header that comes immediately after the IPv6 header. This may be a higher level protocol header, in which case the receiver will use the value of this field to demultiplex the data and forward it to the correct process. Alternatively, it could be an IPv6 extension header for an IPv6 option in which case the receiver will decode and process the extension header. Each extension header also has a Next Header field that can define either a higher level protocol header or another extension header. The receiver therefore has to process all the extension headers until it comes to a Next Header field that represents a higher level protocol, at which point it can then demultiplex and forward the data to the correct process.

6.9.4 Segmentation (Fragmentation)

Fragmentation is handled in a similar way to IPv4 fragmentation, but via an extension header which contains:

- an 8-bit Next Header field
- an 8-bit field reserved for future use
- a 13-bit Fragment Offset field which is measured in 64-bit units
- a 2-bit field reserved for future use
- a 1-bit M flag to indicate that there are more fragments to come or whether it is the last fragment
- a 32-bit Identification field.

6.9.5 Error control

There is no error control function in IPv6. The header has no checksum field. The designers of IPv6 concluded that the error performance of modern networks is such that it is no longer necessary to protect the header against errors.

6.9.6 Flow control

IPv6 will support the ability of routers to signal to hosts explicit congestion notifications in the datagrams that they are forwarding. They do this in both directions by means of Forward and Backward Explicit Congestion Notifications in the Traffic Class field. It will help regulate the transmission of data when congestion occurs without worsening congestion by transmitting ICMP Source Quench packets.

6.9.7 Grade of service

IPv6 addresses grade of service by means of the Diffserv mechanisms which will allow routers to offer differentiated services to different classes of traffic. In addition to this it has the facility to identify traffic flows, such as real-time video, that will require the reservation of resources to meet stringent requirements.

6.9.8 Security

IPv6 has extension headers that can be used for authentication and encrypted security payload thus effectively incorporating the IPsec protocols into IPv6.

6.10 ICMP Version 6 (ICMPv6)[24]

A new version of ICMP has been designed alongside IPv6. It has all the features of ICMPv4 but with some changes to the packet types and codes and some new ones required by IPv6. IPv6 has also incorporated the functions of ARP and IGMP, but not RARP, as it is now seldom used.

[24] *RFC 2463.*

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
    i. For carrying real-time applications, connection-oriented network layer protocols should be used.
    ii. If a datagram exceeds the Maximum Segment Size of the subnetwork, it must be fragmented.
    iii. IPv4 address space is being exhausted because of the high demand for Class A addresses.
    iv. The IPv4 Type of Service field has been redefined to support Differentiated Services.
b. Identify four differences between connection-oriented and connectionless network layer protocols.
c. A company has been allocated a range of IPv4 addresses between 192.16.128.0 and 192.16.135.255. What is the network address that it has been allocated and what is its prefix length and net mask?
d. Describe how IP fragments and reassembles datagrams.
e. Describe how a router is able to resolve an IP address into a physical address on a Local Area Network.

Learning outcomes

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

- describe the services and interfaces offered by the network layer
- describe the functions of the network layer
- explain how the Internet Protocol (v4 and v6) operates
- calculate the range and number of IP host addresses given the network address and the prefix length or net mask and vice versa
- use the ping command and its most common options to determine Round Trip Times to hosts
- use the traceroute command to determine the route to a host
- explain the operation of ICMP and its most common packet types
- describe how IP addresses are resolved into physical addresses and vice versa
- explain how the IPsec protocols operate.

Notes

Chapter 7: The data link layer

In this chapter, we shall examine the services and interfaces provided by the data link layer and the functions of various data link layer protocols. We shall focus on error detection, correction and medium access control mechanisms, as these are the most important functions of the data link layer. We shall examine Binary Synchronous Communications (BiSync / BSC), High-level Data Link Control (HDLC), Point-to-Point Protocol (PPP) and Logical Link Control (LLC) protocols in some detail.

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, (2003). Part 3, Chapters 10, 11, 12, 13, 14.1.

Kurose and Ross *Computer Networking – A Top Down Approach featuring the Internet.* (Addison Wesley), third edition, (2005). Chapters 3.4.1, 3.4.3, 3.4.4, 5.1, 5.2, 5.3, 5.8.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition, (2003). Chapters 5.5.1–2, 6.1, 6.2, 6.3, 6.4, 7,15.3, 16.2.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002). Chapters 3.1, 3.2, 3.4.1, 3.4.2, 3.6, 4.1, 4.2.1, 4.2.2, 4.3.3, 4.3.4, 4,3.9.

7.1 Services

The data link layer, like most other layers, can offer a connection-oriented or connectionless service to its upper layer. You may think that connection-oriented services are only required at the network layer or above, but some data link protocols require some interaction to take place to set up the data link before data can be sent and to close it down afterwards. Some of these protocols also require state information to be held at each end, to support sliding window flow and error control. All of this points towards the need for connection-oriented as well as connectionless operation.

7.2 Interfaces

The interfaces provided by the data link layer must support the usual connectionless and connection-oriented primitives. Again, they are rarely seen by anyone other than network layer protocol developers.

7.3 Functions

The main role of the data link layer is to move data across a single physical link. All the higher layers transmit data across subnetworks which are often made up of multiple data links, but the data link layer is responsible for just one physical link. The physical link can be a private circuit (metallic or optical), an Ethernet bus, a Token Ring or a wireless link. The data link layer must also detect and recover from transmission errors and regulate the flow of data across the physical link.

We will now look at the main functions of the data link layer.

7.3.1 Encapsulation

The data link layer receives data from the network physical layer and encapsulates it, so that the network layer packet is transmitted inside a data link layer frame. Unlike most other layers, the data link layer not only precedes the higher level data with a header, it also uses a trailer for error detection codes which follows after the data. At the receiving end, the data link layer decapsulates the frame, checks it for errors, and if there are none, passes the packet contained within the frame to the network layer.

An important function of the data link layer is the framing of data. Data is received by the data link layer from the network layer in the form of packets. The data link layer must encapsulate these packets into frames for transmission over the physical link and decapsulate the data at the receiving end. In order to do this successfully the data link layer at the receiving end must be able to determine the start and end of each frame.

There are three ways that this can be achieved. Firstly, special bit sequences could be used to indicate the start and end of frames. Secondly, the frame header could contain a length field so that the receiver can determine the end of the frame and then assume that any further data is a new frame. Finally, the start or the end of the frame could be indicated by a code violation. A code violation breaks a rule being used for the coding of data at the physical layer, which can be used by the data link layer to indicate the start or end of a frame. Each method has some problems and data link layer protocols often combine two of the above methods.

Framing can be **character-oriented** or **bit-oriented**. Character-oriented data link protocols transmit data as a sequence of 8-bit characters. Protocol header fields are also often made up of individual characters from the ASCII character set and the data carried must be in multiples of 8 bits. Bit-oriented data link protocols have protocol headers that can code information in binary rather than ASCII formats. For instance, a numeric field which always contains a number less than 8 could be coded as a 3-bit binary number. Data carried can, in theory, not be in multiples of 8 bits, but in practice this would be unusual.

The most common character-oriented data link protocol is the Point-to-Point Protocol used for Internet access.

Bit-oriented data link protocols use a special bit sequence called a flag to synchronise communications and indicate the start and end of frames. Flags are continually transmitted down the link when there is no data to send. This allows the receiver to be kept in synchronisation with the transmitter. In the event of the receiver losing synchronisation, it simply looks for a flag in order to re-synchronise.

One of the problems that can occur with any data link protocol is **transparency**. The protocol should not impose any restrictions on the data to be transmitted. In particular, it must be possible to transmit the special data used to delimit frames and synchronise the receiver (such as flags, and other special characters) within the data itself without confusing the receiver. This is vital, since executable files, which are binary, could easily contain bits that represent these special characters.

Transparency problems are overcome by a procedure called **stuffing** (either byte stuffing for character-oriented protocols or bit-stuffing for bit-oriented protocols). The procedure involves inserting character or a bit in the data stream to prevent the receiver getting confused between codes being used by the data link protocol and codes being transmitted in the data.

Byte stuffing requires using a special Data Link Escape (DLE) character that is stuffed into the frame to indicate that the following character is data. It therefore should not have its normal meaning as a special character such as one that indicates the end of text. If the data includes a DLE, this is also stuffed with another DLE. When a frame is received, any DLEs are unstuffed and the data is passed to the layer above.

Bit stuffing is somewhat similar. Whenever there are five consecutive ones, a single bit (0) is stuffed in the data so that 111110 is sent. The receiver when it receives five consecutives ones followed by a 0 will un-stuff the zero bit and read it as 111111.

For example, if the data link layer wants to transmit 01111110 as transparent data, which could be confused with a flag, it will stuff a zero after the five consecutive ones and send 011111010. The receiver will not interpret this as a flag, but will un-stuff the extra 0 to reproduce the original bit sequence 01111110.

Modern Data Communications mainly uses synchronous transmission, as it is much more efficient and can support sophisticated error control techniques. Asynchronous transmission is usually associated with low-speed (up to 1200 bits/s) modem links. Where such asynchronous transmission exists data can still be transmitted in blocks with error correction using file transfer protocols such as XMODEM, YMODEM, ZMODEM, BLAST and KERMIT.

Bit-oriented synchronous data link layer protocols, such as High-level Data Link Control (HDLC), have tended to replace the older character-oriented data link protocols such as BSC. They are much more sophisticated and efficient as they can support superior flow control and error recovery mechanisms.

7.3.2 Addressing

The data link layer is responsible for the addressing of devices attached to data links. Many data link layer protocols have either been designed to work over a shared medium or can trace their origins back to protocols used in terminal to host network where many terminals shared a data link to a host. In both cases some form of addressing is required and all data link protocols, even those that are only ever used in point-to-point configurations, have address fields even though they are not strictly needed. Point-to-Point Protocol (PPP) commonly used on dial-up Internet connections uses the broadcast address of all 1s for all of its addresses. Addresses on point-to-point and point-to-multipoint circuits only have local significance, but addresses used by data link protocols on LANs are globally unique. They are often called **Media Access Control (MAC)** or **physical addresses** as they identify physical devices, or at least their Network Interface Cards (NICs), onto which they are burned. They have been standardised by the IEEE[1] and are used on all types of LANs. They are 48 bits long and are normally represented in hexadecimal form in pairs separated by hyphens or colons. Each pair corresponds to one byte of the address, as below.

[1] *IEEE 802. 1a.*

00:06:F5:00:1E:2C or 00-06-F5-00-1E-2C

The first three bytes make up the Organisationally Unique Identifier (OUI), which identifies the manufacturer of the NIC. The last three bytes are a serial number assigned by the manufacturer.

7.3.3 Connection control

Some data link layer protocols (such as HDLC and PPP) are connection-oriented with the usual three connection phases (connection, data transfer and disconnection). These protocols require that a link is set up before data can be transmitted which must be closed afterwards. The connection and disconnection of the link is achieved through special frames known as supervisory frames which carry no data. Frames that carry data are called information frames. Connection-oriented data link protocols are able to guarantee the delivery of data in order, without loss and without duplication.

7.3.4 Ordered delivery

Connectionless data link layer protocols will make a best effort to deliver data in the order that it was transmitted, but should an error be detected and a frame discarded, the higher level protocols will be expected to retransmit the data and to reorder it, if necessary. With connection-oriented protocols, the data link layer itself will guarantee ordered delivery and will, if necessary, buffer new data while it is waiting for previous data to be re-transmitted, so that it will always pass the data in the correct order to the network layer. Ordered delivery is achieved through the use of sequence numbers in the data link layer protocol headers.

7.3.5 Flow control

Connectionless data link layer protocols do not support any flow control. If the receiver cannot keep pace with the transmitter, then frames will be discarded and higher level protocols will be required to recover the lost data. Connection-oriented data link layer protocols will usually provide mechanisms to control the flow of data.

There are a number of ways in which flow control can be achieved:

- **X-on/X-off**. This mechanism is only used with asynchronous transmission. Whenever a receiver wants to stop a transmitter, it sends the ASCII DC3 character (Ctrl-S) and when it wants to restart the transmitter it sends the ASCII DC1 character (Ctrl-Q).
- **Stop and wait**. Each frame must be positively acknowledged by the receiver before another frame can be transmitted.
- **Sliding window**. This mechanism allows a transmitter to send a number of frames up to a maximum (called the window size) without any frames being acknowledged. At this point the transmitter has to stop transmitting and wait for some earlier frames to be acknowledged. Once some acknowledgements have been received the transmitter can start sending again, but must stop again as soon as the number of unacknowledged frames reaches the window size. The sliding window mechanism relies on a sequence number field in the data link layer protocol header, so that the receiver can indicate which frames are being acknowledged.

7.3.6 Error control

Most data link protocols provide some for of error detection, although only connection-oriented data link layer protocols tend to support error correction. Errors, as we have seen, are mostly generated at the physical layer and with error-prone media it is important to detect and correct errors at the data link layer. Otherwise they will be forwarded over other data links and will eventually have to be corrected at a higher end-to-end layer. This would be very inefficient.

7.3.6.1 Error detection

Error detection can be achieved by transmitting redundant information along with the data, so that it can be checked for errors:

- **Vertical Redundancy Checks**. VRCs are commonly used in asynchronous communications and add a redundant (parity) bit to each 7-bit ASCII character which is transmitted after the data bits and checks for errors. The parity bit is usually set so that when all 8 bits are added together the sum is even. This is called even parity. It is also possible to do a VRC using odd parity. Traditionally, odd parity was favoured for synchronous transmission and even parity for asynchronous transmission. VRCs can detect any single bit error and 50% of burst errors. VRCs can be implemented easily in hardware using exclusive OR (XOR) logic gates.

  XOR is a binary operation, often symbolised by $\oplus$, which was covered in CIS102. To recap, the XOR operation is defined by the following truth table:

Table 7.1: Truth table for XOR operation

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

  If an ASCII character is represented by the following bits b1 to b8, then the parity bit can be calculated from:

  $b8 = b7\oplus(b6\oplus(b5\oplus(b4\oplus(b3\oplus(b2\oplus b1)))))$ and the parity can be checked by evaluating $b8\oplus(b7\oplus(b6\oplus(b5\oplus(b4\oplus(b3\oplus(b2\oplus b1))))))$ which should evaluate to zero.

- **Longitudinal Redundancy Checks**. LRCs are used in conjunction with VRCs to provide a more sophisticated check on a block of data. This is done by treating the block as a two-dimensional array of bits, with each byte transmitted represented by a column in the array, the last bit in each column being a VRC for the column. A parity check (called an LRC) is then carried out for each row in the array, including the VRC row, and a new LRC column is added to the block. The block is then transmitted one byte at a time followed by the LRC byte. On receipt of the block, the data link layer will check all the VRC and LRC bits. If any of these bits does not check then the frame has been damaged during transmission. LRCs on a block of n bytes will be able to detect burst errors of up to n bits long. There are however combinations of errors that LRCs will not be able to detect, such as when single bit errors occur in the same positions of two columns or two rows. LRCs nonetheless provide a powerful error detection mechanism. Table 7.2 shows an example of an even parity LRC check on a 11-byte block. There is one VRC parity error and one LRC parity error. These are underlined and they can be used to pinpoint and correct an error (if it is a single bit error), as well as detect most multiple errors.

Table 7.2: Example of an even parity Longitudinal Redundancy Check

Direction of Transmission of Bits (LRC byte last)

Direction of Transmission of Bits (VRC bit last)

0	1	0	0	1	0	1	1	0	0	1	1
0	0	0	0	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	0	0	1	**1**
1	0	1	1	1	0	0	1	0	1	0	0
0	0	0	0	0	0	1	0	1	0	0	0
0	1	1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	0
0	0	0	0	0	1	**1**	0	0	0	0	0

Cyclical Redundancy Checks. CRCs offer a more sophisticated and powerful error detection technique. The bits of the data to be checked are treated as a very large number which is divided (modulo two[2]) by a special number, known as a generator.

[2] *You may wonder why modulo 2 division is used in CRC checksums. The reason is that it is much easier and more efficient to implement in hardware than normal division. The logic circuits required to do a CRC checksum only require a few shift registers and XOR gates.*

In modulo 2 arithmetic, all additions and subtractions are replaced by the ⊕ (XOR) operation.

As an example of modulo 2 multiplication, we will multiply 1011 by 101 using ⊕ instead of + to add the two partial multiplications:

```
      1 0 1 1
x       1 0 1
-------------
      1 0 1 1
⊕ 1 0 1 1 0 0
-------------
  1 0 0 1 1 1
-------------
```

Modulo 2 division is similar to ordinary binary long division, but subtraction is replaced by the ⊕ operation and the result of a partial division is 1, only if the number being divided starts with a 1 (even if the number being divided is less

```
                1 0 1 1
           ------------
1 0 1  )1 0 0 1 1 1
       ⊕1 0 1
        -----
          0 1 1
        ⊕ 0 0 0
          -----
            1 1 1
          ⊕ 1 0 1
            -----
              1 0 1
            ⊕ 1 0 1
              -----
                0 0
```

than the divisor). Note that at after each ⊕ operation, the most significant bit is dropped and if the next bit is a 0, then a 0 is entered in the quotient on the top line and 000 is used with the ⊕ operation rather than the divisor.

```
                  0 1 0 1 0
             --------------------
1 0 0 1  ) 0 1 0 1 1 0 0 0
         ⊕ 0 0 0 0
           -------
             1 0 1 1
           ⊕ 1 0 0 1
             -------
               0 1 0 0
             ⊕ 0 0 0 0
               -------
                 1 0 0 0
               ⊕ 1 0 0 1
                 -------
                   0 0 1 0
                 ⊕ 0 0 0 0
                   -------
                     0 1 0
```

The remainder of this division is appended to the data as a checksum field in the data link protocol trailer. On receiving a block with a CRC, the receiver will divide the data plus the checksum by the generator (modulo 2) and if there is a remainder, it will know that error(s) have occurred.

CRC ecample 1

We shall consider a simple example where a 5-bit code 01011 being protected by a 3-bit CRC using 1001 as the generator. Firstly, we must add the 3 bits required for the CRC to the data and set each of these bits to zero. Then we divide this by 1001 in modulo 2.

The remainder of this division 010 is appended to the original data and is transmitted as 01011010.

When the data plus checksum is received, this is then checked by dividing it by the generator and if no corruption has taken place, the remainder will be 0. The CRC digits can then be discarded and the receiver knows that the data 01011 has been received without any corruption.

```
                0 1 0 1 0
1 0 0 1 ) 0 1 0 1 1 0 1 0
        ⊕ 0 0 0 0
            1 0 1 1
          ⊕ 1 0 0 1
              0 1 0 0
            ⊕ 0 0 0 0
                1 0 0 1
              ⊕ 1 0 0 1
                  0 0 0 0
                ⊕ 0 0 0 0
                    0 0 0
```

The generators are often expressed as polynomials rather than as binary numbers. The generator 1001 that we used is described as $x^3 + 1$. This is obtained by looking at the power of two that each 1 represents and replacing it with the power of a dummy variable x in a polynomial. The use of polynomials enables mathematicians to develop CRC algorithms and prove their validity and effectiveness. The polynomials are chosen to meet the following criteria:

- It should not be divisible by x (in other words it must end with + 1).
- It should be divisible by (x + 1) modulo 2.

If these conditions are met, then all burst errors that affect an odd number of bits can be detected and all other burst errors less than or equal to the degree of the polynomial (i.e. the length of the CRC field) can be detected and there is a high probability that burst errors greater than the degree of the polynomial will be detected.

There are various different CRC algorithms, using different generators. The choice of which to use depends on the length of the data that needs to be protected, the performance required and the amount of CRC overhead that is acceptable.

CRC-8[3] uses the polynomial $x^8 + x^2 + x + 1$ (or 100000111 as its divisor). It is used to protect ATM headers.

CRC-16 uses the polynomial $x^{16} + x^{12} + x^5 + 1$ (or 10001000000100001 as its divisor). It is used by HDLC.

CRC-32 uses a very complex polynomial with 15 terms. It is used by LAN Media Access Control protocols.

[3] The number after the CRC- indicates the degree of polynomial which is also the length of the CRC field in bits.

The CRC-8 checksum is therefore one byte long, the CRC-16 checksum is two bytes long and the CRC-32 checksum, four bytes long. CRC-16 and CRC-32 are most commonly used. CRC-16 will detect 100% of odd burst errors and all even burst errors of 16 bits or less and 99.997% of even burst errors longer than 16 bits. CRC-32 will detect 100% of odd burst errors and all even burst errors of 32 bits or less and 99.99999998% of even burst errors longer than 32 bits. It is therefore very improbable that CRCs will not detect errors that have occurred transmitting a block of data.

Activity 7.1

Choose a character from the ASCII character set and look up its ASCII 7-bit code in a table. Verify that its even parity bit can be calculated using the XOR functions given above and that the parity of the 8-bit code can also be checked using the given XOR functions.

Activity 7.2

Using the generator 10111, Calculate the 4-bit CRC check on the ASCII character you chose above. Write out the character appended by its 4-bit CRC checksum and use the same generator to perform a CRC to prove that there are no errors. Alter any combination of bits in the 12-bit code and verify that the CRC will detect the error(s).

7.3.6.2 Error correction

In the early days of data communications, when low-speed asynchronous duplex data links were used between terminals and host computers, a crude error detection technique called **echoplexing** was used. Characters were typed at a terminal and were echoed back by the host. It was the echoed character and not the sent character that was displayed to the user. In conjunction with echoplexing, VRC checks were also used and any character that failed a VRC check was displayed as a block. By this means users were able to see errors and re-enter the data. This echoplexing technique is still sometimes in use today. Many Telnet applications echo back characters to users, often with packets containing single characters.

There are two main error correction techniques used by modern data link protocols:

- **Automatic Repeat Request (ARQ)**
- **Forward Error Correction (FEC).**

With **Automatic Repeat Request (ARQ)**, the receiver issues a Negative Acknowledgement (NAK) or refrains from issuing a Positive Acknowledgement (ACK) when an error is detected, thus expecting the transmitter to retransmit the frame (possibly after a timeout has expired) and maybe other frames also.

The number of frames that will need to be transmitted after an error depends on the precise error correction mechanism employed.

The mechanisms used by data link protocols are:

- **Idle RQ** sometimes also called **Stop and Wait**, where the sender waits for a positive acknowledgment before it can send another frame.
- **Continuous RQ** which uses the same sliding window technique discussed under flow control to allowing a fixed number[4] of frames to be unacknowledged at any one time.

  With **Continuous RQ** there are two possible mechanisms that can be used:

  - **Go Back N** where on detection of an error, the receiver issues a NAK and discards all frames that it received subsequent to the error. The

[4] *The window size.*

transmitter then has to retransmit the frame that was in error and all subsequent frames.

- **Selective Repeat** where on detection of an error, the receiver issues a Reject (REJ) frame, but keeps all the subsequent frames in buffer. The transmitter only retransmits the frame that was in error.

It would seem that Selective Repeat is the more efficient scheme as frames which were received without errors are not retransmitted unnecessarily. But many data link protocol implementations choose to use Go Back N rather than Selective Repeat. This is because of the complexity of the code that must be written to buffer all the good frames and reorder them when the bad frame is received successfully after retransmission. Also, with modern transmission systems, errors are relatively rare and putting a lot of work into writing complex code that is not used very much does not make sense.

With **Forward Error Correction** (**FEC**), redundant information (much more than is needed for error detection alone) is sent along with the data, so that errors can not only be detected but, in many cases, corrected as well. These schemes are by no means foolproof and protocols that use FEC will also have to implement ARQ for situations where errors are detected but cannot be corrected. We have already seen how the LRC can pinpoint and correct a single bit error. Another well known FEC scheme which can be extended to correct multiple bit errors is the use of Hamming codes[5]. The Hamming code can be used to correct a single bit error in an arbitrarily long sequence of bits (but often 8 bits).

[5] *Named after its inventor, Richard Hamming of Bell Labs.*

The Hamming code uses multiple parity bits within a code, where each parity bit checks a unique combination of bits. The parity used can be either even or odd.

The parity bits appear in the bit positions which are powers of two (bits 1,2,4,8, etc.). The remaining bit positions are occupied by the bits that are to be checked. The parity bits check the data bits according to the following table:

Table 7.3: Hamming code parity checks

b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	checked by
1	0	1	0	1	0	1	0	1	0	1	0	**b1**
0	1	1	0	0	1	1	0	0	1	1	0	**b2**
0	0	0	1	1	1	1	0	0	0	0	1	**b4**
0	0	0	0	0	0	0	1	1	1	1	1	**b8**

Therefore, bit1 checks all the odd bits, bit2 checks b2, b3, b6, b7, b10, b11, etc.

Note also that the binary code for the bit number at the top of each column can be obtained by reading all of the bits in that column from the bottom up.

Hamming code example 1

We will now look at an example of how to code the byte 01101110 in an even Hamming code. We place the bits from this byte in the first row of a table headed with the bit numbers in reverse order,[6] leaving spaces for the parity bits. In the b1 row, we place all the bits that are checked by b1 and then calculate the value of b1 which will give even parity and enter it in the b1 column. We do the same to calculate b2 in the b2 row, b3 in the b4 row and b8 in the b8 row.

[6] *In this subject guide, and in most textbooks, bit1 is assumed to be the low order bit that is placed in the rightmost position. Tanenbaum (pages 194–5) for some reason assumes the opposite.*

Table 7.4: Encoding 01101110 into an even Hamming code

	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
	0	1	1	0		1	1	1		0		
b1		1		0		1		1		0		**1**
b2		1	1			1	1			0	**0**	
b4	0					1	1	1	**1**			
b8	0	1	1	0	**0**							

Now we have calculated all the parity bits and hence the even Hamming code for 01101110 is therefore 011001111001. This code is then transmitted.

We will now see what happens when one bit in this bit string is corrupted, so that the bit string 011000111001 is received. We enter the data received in the first row. We check the parity calculations for b1, b2, b4 and b8. If the total parity is odd, we underline the parity bit and place a 1 in the 'fix' column and a 0 otherwise, which is what must be added to ensure even parity.

Table 7.5: Finding an error when an even Hamming code 011000111001 is received

	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	fix
	0	1	1	0	0	0	1	1	1	0	0	1	
b1		1		0		0		1		0		**1**	1
b2		1	1			0	1			0	**0**		1
b4	0					0	1	1	**1**				1
b8	0	1	1	0	0								0

We can see that the error occurred in parity bits 1, 2 and 4 but not in 8. The b7 bit is the only one that is checked by bit1, bit2 and bit4 and nothing else, which means that it is therefore the one that has been corrupted. As an additional check, we can read the bits in the 'fix' column upwards from the bottom and converting the binary number to decimal gives 7 confirming that the error was in b7. The bit error can now be corrected and the original byte that was sent was 11001101.

Unfortunately, most transmission errors are caused by impulse noise, which will cause a burst of errors rather than a single bit error. Hamming codes can be extended to correct burst errors of up to N bits in a block of N bytes (where N is an arbitrary number), by changing the order of bit transmission. If a sequence of bytes is Hamming coded and then assembled into a block of N bytes and the first bit of each Hamming code in the block is transmitted in turn followed by the second bit of each Hamming code, then the third etc., until the 12th bit of the Nth Hamming code is transmitted, then a single burst error of up to N bits can be corrected. This is illustrated in Table 7.6 which shows four bytes (a, b, c and d) encoded as Hamming codes assembled into a block for transmission.

Table 7.6: Example to show how a single 4-bit burst error can be corrected using Hamming codes.

a12	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1
b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
c12	c11	c10	c9	c8	c7	c6	c5	c4	c3	c2	c1
d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1

These would normally be transmitted as:

a1,a2,…,a12,b1,b2,…b12,c1,c2,..c12,d1,d2,…d12

But if the codes are transmitted in the order:

a1,b1,c1,d1,a2,b2,c2,d2,…a12,b12,c12,d12

then any single burst error of up to 4 bits will result in at most a single bit error in each row. So when the data is received and reordered into the four Hamming codes, there can only be at most one error in each Hamming code, which can then be corrected.

Hamming codes provide a useful technique for forward error correction on data links that are very error-prone such as radio channels, but there is a price to pay in overheads. The amount of redundant data transmitted to provide FEC on 8-bit characters using Hamming codes adds an overhead of 50%, but there is a potential saving in that retransmissions are reduced.

Activity 7.3

Study the Go Back N and Selective Repeat ARQ mechanisms in a textbook, or using a web site. Satisfy yourself, by drawing timing diagrams (showing sequence numbers and acknowledgement numbers), that the two mechanisms employed between the A-end and the B-end of a data link can recover from the following problems:

a. the corruption of a frame transmitted from A to B
b. the duplication of a frame transmitted from A to B
d. the corruption of an acknowledgement transmitted from B to A
e. the loss of an acknowledgement transmitted from B to A
f. the duplication of an acknowledgement transmitted from B to A.

Activity 7.4

Examine the data link interface statistics on your computer to check the ratio of errors and discards to successfully transmitted frames. The following command lines can be used to obtain data link interface statistics for DOS and Unix respectively.

DOS: netstat -e Unix: netstat -i

Activity 7.5

Binary Coded Decimal is a code that is used to represent the decimal digits 0–9 as 4-bit binary codes, by simply coding each decimal digit as its 4-bit binary conversion (e.g. 7 is encoded as 0111). Add error checking bits to each of the 10 BCD codes to create even Hamming codes, so that any single bit error in a BCD code can be corrected. Choose one BCD code and introduce a single bit error into each bit of its Hamming code in turn (including the parity bits) and show how the error can be detected and corrected.

7.3.7 Link Control

Connection-oriented data link protocols have to support frame types other than the **information frames** that carry data. Additional frames types are required for control purposes. These frames are usually called **supervisory frames** if they contain sequence numbers and **unnumbered frames** if they do not. The supervisory frames support positive and negative acknowledgements for flow and error control. The unnumbered frames perform functions such as setting up, resetting and disconnecting the data link.

Connectionless data link protocols only require unnumbered information frames.

The data link layer protocols used on LANs are defined in two sub-layers. The upper sub-layer is the **Logical Link Control** (**LLC**) sub-layer and is common to all types of LAN.

7.3.8 Media Access Control

The lower sub-layer is called the **Media Access Control** (**MAC**) sub-layer and is dependent on the type of LAN. The IEEE is also responsible for specifying the standards for the different MAC sub-layer protocols. The MAC sub-layer is responsible for controlling access to the transmission medium in such a way that capacity is allocated orderly and efficiently. It is also responsible for framing, addressing and error checking.

The physical medium at the level below the data link layer can be one of three basic types. It can be:

- Point-to-Point – a normal telecommunications circuit with two ends.
- Point-to-Multipoint – a multidrop circuit in a tree topology supporting a small number of devices (or **stations**) which can each view all the transmissions on the medium. This type of circuit was very popular with mainframe star networks, where circuits were configured with the host at one end and multiple terminals in different locations at the other ends.
- Broadcast – a shared medium usually in a bus or ring topology or using wireless transmission, where any number of stations, usually of equal status, can view all the transmissions on the medium.

In each of these types there can be a problem if two stations are able to transmit at the same time. This can obviously happen with broadcast and point-to-multipoint media but it can also happen with a point-to-point medium, if it is configured in half-duplex mode.

It is the responsibility of the data link layer to either prevent stations from transmitting at the same time or else to resolve the contention for the medium should it occur.

There are three main types of medium access control methods that the data link layer can employ. This classification is taken from Forouzan.[7]

[7] Forouzan, Chapter 13, pages 311–325.

- **Random Access** is where any station can transmit at any time and the data link protocol subsequently resolves any contention.
- **Controlled Access** is where the data link protocol provides an authorisation to transmit.
- **Chanelised Access** is where the data link protocol allocates a separate channel to each transmitting device.

Random Access, where there are few, if any, constraints on when a station can transmit, is the simplest of all access methods. However, the data link protocol must be able to detect when stations are transmitting simultaneously (a **collision**) and have a procedure to resolve the contention. There are four different access methods that can be employed:

- **Multiple Access (MA)** is the simplest random access method. There is absolutely no constraint on when to transmit. A transmitter can even transmit after another station has already started to transmit. The original ALOHA packet radio system used this method for data communication between the Hawaiian Islands. After sending a frame the station has to wait for an acknowledgement. If this is not received it retransmits the frame after a random time (this procedure is called **backoff**).
- **Carrier Sense Multiple Access (CSMA)** is a refinement of Multiple Access where the transmitter listens on the energy levels in the medium to check that no other station is transmitting, before beginning to transmit itself. It therefore senses the carrier before transmitting. This method reduces the possibility of collisions, but they can still occur, since

there is always a propagation delay on the medium. This means that a station could start transmitting after it has sensed that the medium is free, but could be unaware that another station has already started to transmit. If, when sensing the medium, it is found to be busy, CSMA systems can implement persistence strategies to decide what to do next. With a **non-persistent strategy**, the station will implement the backoff algorithm. In a **1-persistent strategy**, the station will start transmitting (with a probability of 1), as soon it detects the medium is free, but so could other stations. In a **p-persistent strategy**, the station will start transmitting immediately with a probability of p and will back off with a probability of 1-p. Ethernet uses a 1-persistent strategy. CSMA does not specify what a station should do if it senses that a collision has occured. There are two choices:

- **Carrier Sense Multiple Access with Collision Detect (CSMA/CD)** implements a backoff algorithm when a station senses that a collision has occurred. Once it starts transmitting, it also continues to listen to the medium to sense any collisions. If it detects a collision during transmission, it immediately backs off. CSMA/CD is the access method used by Ethernet LANs.
- **Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)** makes no attempt to detect collisions. With some media, e.g. radio, it is expensive to provide equipment that can transmit and receive at the same time. CSMA/CA avoids this extra cost by firstly implementing one of the persistence strategies and then waiting a set period of time, called the Interframe Gap (IFG), after which it waits for another random period of time and then transmits. It then switches to receive mode, starts a timer and awaits an acknowledgement. If the station does not receive an acknowledgement before the timer expires it backs off and retransmits. CSMA/CA is used in wireless LANs.

Controlled Access is where one station (the primary station) assumes the responsibility for controlling access to the channel. There are three types of controlled access:

- **Polling Access Method**,[8] which was the method used by mainframe computers where they issue a request to each of the secondary stations in turn for them to send data. This technique can, however, also be used on a point-to-point circuit between two peer devices, one of which must assume the primary station role. The primary station, when it wants to send, issues Select Frames and polls each of the secondary devices in turn to request any frames that they might want to transmit. The primary station must await a response from the secondary station which can either be an Information Frame or an NAK and if the latter the primary station must respond with an ACK. It can then move on and either issue another select frame or poll another secondary station.
- **Reservation Access Method**[9] is where each station needs to make a reservation before transmitting. This is achieved by allocating each station a reservation **minislot** within a reservation frame sent out by the primary station. Stations indicate that they want to reserve a timeslot by indicating this in their own minislots in the reservation frame. After the reservation frame has been transmitted, each station that has made a reservation can send a frame in turn, after which another reservation frame is sent. This method can be used for cable modem and satellite networks.
- **Token Passing Access Method**[10] is where a special bit pattern (called a **token**) is circulated to each of the stations in turn and if a station wants to

[8] *Used in host-based star topology computer networks.*

[9] *Used for resolving contention in DOCSIS cable modems.*

[10] *Used in IEEE 802.5 Token Ring LANs.*

transmit it has to wait until it receives the token. It then captures the token, transmits its frame and then releases the token so that another station can transmit. Token passing is used in the Token Ring and Fibre Distributed Data Interface (FDDI) LAN protocols.

Channelised Access is a method whereby the available bandwidth of the media is shared between different stations by allocating a channel to each transmitter. There are three ways that this can be achieved:

- **Frequency Division Multiple Access (FDMA)**[11] – where each transmitter is allocated a different frequency range on the medium in which to transmit. It uses Frequency Division Multiplexing at the physical layer.
- **Time Division Multiple Access (TDMA)**[12] – where each transmitter is allocated different time slots on the medium in which to transmit. It uses Time Division Multiplexing at the physical layer.
- **Code Division Multiple Access (CDMA)**[13] – is a new technology, based on coding theory, which differs from FDMA in that the media only supports one broad channel and from TDMA in that stations do not have to use different timeslots. Each transmitter is allocated a unique, carefully selected code (or chipping sequence), and they can remarkably actually transmit simultaneously and only interfere with each other minimally. It is a bit like holding conversations at a very noisy party. The details of how CDMA works are beyond the scope of this course.

[11] *Used in first generation mobile phone networks and still used in satellite communications.*

[12] *Used in second generation mobile phone networks and also in satellite communications.*

[13] *Used in third generation mobile phone networks and in IEEE 802.11 wireless LANs.*

7.4 Binary Synchronous Communications (BSC or BiSync)

Bisync is an old protocol developed by IBM and standardised by ISO. It is a connection-oriented half-duplex, asymmetric, synchronous, character-oriented, byte-stuffed protocol, using the Data Link Escape (DLE) character for data transparency. It uses a polling access method and it can operate in point-to-point or point-to-multipoint line configurations.

It uses two or more ASCII Synchronisation characters (SYN) to synchronise the receiver for the start of a frame, followed by an optional ASCII Start of Header character (SOH) to indicate the start of a frame header, Start of Text (STX) for the start of the data field and End of Text (ETX) for the end of the data field with an optional one or two character Block Check Count (BCC) to provide error detection using the LRC technique. It uses Idle RQ (Stop and Wait) error control.

Below are examples of typical BSC frames:

SYN SYN STX <Data> ETX

SYN SYN SOH <Header> STX <Data> ETX <Block Check Count>

Bisync was mainly used in star-topology mainframe networks, but has now been largely replaced by modern bit-oriented protocols, such as HDLC.

7.5 High-level Data Link Control (HDLC)

HDLC was developed by ISO but was based largely on an IBM bit-oriented protocol called SDLC. It is a connection-oriented full-duplex, synchronous, bit-oriented, bit-stuffed protocol, using a polling access method. It uses CRC error detection and can support both Go Back N and Selective Repeat error control. It is a complex protocol with many implementation options. HDLC uses an 8-bit flag (01111110) for synchronisation purposes at the start and end of each frame.

HDLC has two classes of operation.

- **Normal Response Mode** (**NRM**), which can operate in point-to-point or point-to-multipoint line configurations. In this mode, it is asymmetric with one station acting as a master and the other station(s) as slave(s). It supports Continuous Go Back N ARQ error control. There are no explicit negative acknowledgements. NRM is often used in star topology mainframe networks.
- **Asynchronous Balanced Mode** (**ABM**), which can only operate over point-to-point line configurations. It is a symmetric protocol (both stations having equal status). It does use explicit negative acknowledgements (REJ for Go Back N and SREJ for Selective Repeat). ABM is used as a data link protocol on many private circuits (up to 2 Mbit/s), and is often used as the data link protocol on private IP networks. A commonly used subset of ABM is Link Access Protocol Balanced (LAPB). It only supports Go Back N error control, but it does support extended window sizes of up to 128, which is useful for high-speed circuits and satellite circuits. LAPB is the data link protocol used on X.25 packet switched networks.

7.6 Point-to-Point Protocol (PPP)[14]

[14] RFC 1661.

PPP is a suite of protocols designed to carry IP over modem links. Access to Internet Service Providers normally uses PPP as the data link protocol, for both dial-up and broadband connections. It supports multiple network layer protocols, besides IP, including AppleTalk and IPX. As its name implies, it only supports point-to-point line configurations. It is a connection-oriented, full-duplex, symmetric, synchronous, character oriented, byte stuffed protocol. It uses CRC-16 or CRC-32 error detection, but has no flow control or error recovery mechanisms. It relies on higher layer protocols to recover from errors.

PPP uses a 1-byte flag (the same as HDLC's) for synchronisation at the start and end of each frame. The header consists of a 1-byte Address field which is always set to the broadcast address (all ones), a Control field that defines the type of frame, a protocol field for demultiplexing the various protocols that PPP can carry, a variable length data field, followed by a 2 or 4-byte CRC field and then another flag.

PPP frames carry within them other protocols from the PPP suite:

- **Link Control Protocol** (**LCP**) is used to establish, configure, maintain and close down the data link. When establishing a link, LCP is able to negotiate options such as the Maximum Receive Unit that can be handled and which error correction techniques or data compression algorithms, if any, will be used. LCP is optionally able to determine the quality of the link and close it down, if it is of poor quality. LCP packets do not carry any user data.
- The authentication protocol to be used to identify the user to the ISP is one of the options that LCP negotiates when it establishes a link. There are two alternative authentication protocols supported by PPP:
  - **Password Authentication Protocol** (**PAP**) is a simple authentication protocol that sends user name and passwords in plain text. PAP is only used at link establishment.
  - **Challenge Handshake Authentication Protocol** (**CHAP**) is a much more secure form of authentication using a three-way handshake. The password is kept secret and the client may from time to time (not just at link establishment) be sent a challenge value. It uses a predefined function that takes the challenge value and the user's password as its

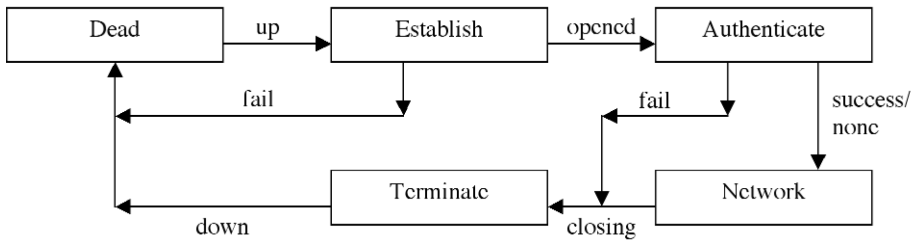
input to produce a result that is transmitted back to the challenger. The challenger then issues a packet to accept or reject the authentication.

- **Network Control Protocols (NCPs)** which are used to configure the various network layers supported by PPP. There is therefore a different NCP for most protocols that PPP supports. IP Control Protocol (IPCP)[15] is used to set up the IP network layer (including the configuration of the IP address).

[15] *RFC 1332.*

As a connection-oriented protocol, PPP has a number of phases through which it must go through to establish and close connections. Figure 7.1 shows the simplified phase diagram for PPP.

Figure 7.1: Simplified Phase Transition Diagram for PPP



The Dead phase is where no connection exists. In the Establish phase, the LCP establishes the data link and negotiates the data link options before entering the Authenticate phase where PAP or CHAP are used to authenticate the user. This phase may be bypassed if no authentication is required. In the Network phase, the NCP (we will assume IPCP in this case) negotiates and configures the network layer options and assigns the IP address. Once this is done, IP packets can then be encapsulated by PPP. When the data link is no longer required, IPCP will terminate the network layer activity and LCP will enter the Terminate phase and once the data link is closed, it will return to the Dead phase.

7.7 Logical Link Control (LLC)[16]

[16] *IEEE 802.2.*

The LLC sub-layer has been standardised by the IEEE and is common to all LAN protocols. It has the following fields:

- A 1-byte **Destination Service Access Point** field, which specifies the destination user of LLC for demultiplexing. The first bit indicates whether it is an individual or group address.
- A 1-byte **Source Service Access Point** field, which specifies the destination user of LLC. The first bit indicates whether the PDU is a command or a response.
- A 1- or 2-Byte **LLC Control** field, which is identical to that used in HDLC with extended (7-bit) sequence numbers.

The LLC standard defines three types of service that can be offered using the same PDU formats:

- LLC Type 1 (LLC1) – unacknowledged connectionless service
- LLC Type 2 (LLC2) – acknowledged connection-oriented service
- LLC Type 3 (LLC3) – acknowledged connectionless service.

Of these LLC types, LLC1 is most commonly used, as acknowledgements are usually provided by higher layer protocols rather than at the data link layer.

The LLC protocol makes use of unnumbered HDLC frame types to support LLC1, ABM frame types to support LLC2 and two new unnumbered frame types to support LLC3 acknowledgements.

LLC supports multiplexing by means of Source and Destination Service Access Points which are 7-bit addresses used to identify the users of the LLC service.

7.8 Ethernet Media Access Control (IEEE 802.3)[17]

[17] IEEE 802.3 is the most common Ethernet MAC protocol. An earlier protocol called Ethernet II is still in use however.

We will study the Ethernet MAC protocol in this part of the course, as it is a common method of interfacing a personal computer to a broadband modem (ADSL or cable) which will be covered in the next chapter. Other LAN MAC protocols will be studied in the Enterprise Networking part of the course, as they are most commonly used by enterprises.

The Medium Access Control for Ethernet uses the CSMA/CD multiple access technique. The backoff algorithm used by Ethernet is the **Binary Exponential Backoff Algorithm**. Each transmitter chooses a random number between 0 and $2^N - 1$, where N is the number of times the frame has suffered from a collision. This is then multiplied by the maximum round trip time that it can take a frame to travel from one end of the LAN to the other end and back. The station then waits for this time, before it retransmits the frame. Thus in a very heavily congested LAN that suffers from successive collisions, the spread of backoff times will grow exponentially after each collision. In a lightly congested LAN, the backoff times are likely to be quite small, as there is a high probability that the stations will select different backoff times. After 10 collisions the backoff time is frozen, and after 16 collisions, the algorithm gives up and reports failure.

The Ethernet 802.3 MAC frame[18] consists of the following fields:

[18] Ethernet II has a 64-bit Preamble and no Start of Frame Delimiter. It does not use LLC, so it needs a protocol field for demultiplexing. The Length field is replaced by a 16-bit Type field.

- A 7-byte **Preamble** of alternating 1s and 0s. Strictly speaking, it is not part of the data link frame. It is added by the physical layer. Its purpose is to synchronise the receiver before the start of the frame.
- A 1-byte **Start of Frame Delimiter** (10101011) to alert the receiver that a frame is beginning.
- A 6-byte **Destination Address** field.
- A 6-byte **Source Address** field.
- A 2-byte **Length/Type**[19] field to indicate the length of the data field in bytes or its Ethernet II type.
- A variable length **Data** field (between 0 and 1500 bytes).
- A variable length **Pad** field (between 0 and 46 bytes). All valid Ethernet frames must be at least 64 bytes long. This is to ensure that all collisions are detected. A short frame could be transmitted and collide with another frame, but the transmitter will not detect a collision that has corrupted the frame. If the data being carried is less than 46 bytes then padding is added to make the frame 64 bytes long.
- A 4-byte **Frame Checking Sequence** using CRC-32.

[19] For compatibility with Ethernet II, 802.3 supports both the 802.3 and the Ethernet II interpretations of this field are supported. All of the values of the Ethernet II Type field are greater than 1500, so there is no ambiguity regardng the meaning of the field.

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
   i. LAN Physical addresses are standardised by the IEEE and are 16 bits long. The first 8 bits indicate the card manufacturer and the last 8 bits represent a serial number.
   ii. Ethernet uses the CSMA/CA media access method.
   iii. PPP addresses are always set to the broadcast address of all 1s.
   iv. All Ethernet frames must be at least 64 bytes long. If there is not enough data in the frame to ensure this, they will be padded with extra data.

b. Compare the Go Back N and the Selective Repeat ARQ methods. Which is most commonly used and why?

c. Show how the byte 10101001 can be encoded using an even Hamming code. Another even Hamming coded byte was received with one bit corrupted and the bits received were 110001100101. Show how a one bit error can be detected and then corrected. What was the original byte transmitted?

d. Compare the two alternative authentication protocols supported by PPP.

e. What is the name of the backoff algorithm used on Ethernets. Describe how it works.

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 data link layer
- describe the functions of the data link layer
- demonstrate how Vertical and Longitudinal Redundancy Checks work
- demonstrate how a simple Cyclical Redundancy Check works
- describe the Idle RQ and the two ARQ error correction techniques
- demonstrate how single bit errors can be detected using Hamming codes
- classify different Medium Access Control techniques
- distinguish between bit-oriented and character-oriented data link protocols with particular reference to data transparency
- outline how PPP works
- outline how the LLC protocol works.

Chapter 8: The physical layer

Introduction

In this chapter, we shall examine the services and interfaces provided by the physical layer and the functions of various physical layer interfaces and protocols, including the basic physics of transmission, the physical laws that relate capacity and bandwidth and the various modulation and line coding techniques used. We shall examine in some detail dial-up modems that support narrowband data services, and Asynchronous Digital Subscriber Line (ADSL) and cable modems that support broadband services.

Further reading

Forouzan, Behrouz, A. *Data Communications and Networking.* (McGraw Hill), third edition, 2004. Part 2, Chapters 2, 3, 4, 5, 7.

Stallings, William *Data and Computer Communications.* (Prentice Hall International), seventh edition, 2003. Chapters 3.1, 3.4, 4.1, 4.2, 4.3, 4.4, 5.

Tanenbaum, Andrew S. *Computer Networks*. (Prentice Hall International), fourth edition, (2002) [ISBN 010384887]. Chapters 2.1, 2.2, 2.3, 2.4, 2.5.3, 2.5.4, 2.7.

8.1 Services

The physical layer offers a bit transmission service to the data link layer, so that bits can be transferred serially across a physical medium. Note that it is bits that are transferred by the physical layer and not frames, and although some physical media support multiplexing by means of a frame structure, the physical medium is primarily concerned with the transmission of bits. The services provided are very much determined by the characteristics of the medium.

The physical layer specifies the mechanical, electrical, functional and procedural aspects of bit transmission.

8.2 Interfaces

Both the data link layer and the physical layer on a computer are usually implemented within hardware, such as a Network Interface Card (NIC) for LAN access, a Universal Asynchronous Receiver Transmitter (UART) chip or a Universal Serial Bus (USB) for serial communications with peripherals including modems. Because both layers are supported in hardware, the identification of, and access to, the actual interface between the two layers is problematic. Although there must be a physical layer residing in each computer, this layer usually controls a serial port that is often connected via a short cable to another physical device such as a modem, which is controlled from the port. It is this interface at the serial port that is often considered as the physical interface, but it is really an interface between the physical layer in the computer and the physical layer in the modem. The modem then has another physical interface to the physical medium that carries the bits over a telecommunications circuit. A physical link is, in practice, made up of a series of physical media interconnected with physical layer devices such as multiplexers and repeaters. Sometimes the modem is an internal card in one of the expansion slots of the computer. In this case the physical interface of the computer is actually a telephone line interface.

An example of a computer to modem interface is the ITU-T's V.24 standard, popularly known as RS232.[1] This interface was originally designed to enable a computer to control a modem, so that it could transmit data over a standard telephone circuit. Although the standard was intended for use with modems, it was also used with many other devices, such as serial printers or scanners.

[1] *The RS232 standard is now properly called EIA/TIA 232-F, but it is often still referred to as RS232.*

Another computer to physical layer device interface is the ITU-T's X.21 standard. This interface was designed to allow a computer to connect to a circuit-switched data network via a digital synchronous interface, but it was also chosen as the physical layer for the X.25 standard that allows computers to connect to packet-switched data networks. Many of the control functions supported by V.24 are implemented in X.21 by means of the exchange of encoded data, so an X.21 interface does not need as many pins as a full V.24 interface.

A modern interface standard that is sometimes used for modems (including ADSL modems) is the **Universal Serial Bus** (**USB**). USB has a maximum data rate of 12 Mbit/s and can support up to 127 devices on the same external bus, which consists of four wires. It can support multiple devices at 1.5 Mbit/s or one device at 12 Mbit/s. Multiple devices can be daisy-chained by being connected to other USB devices or to USB hubs. USB also distributes low voltage power to peripherals which no longer require separate low voltage power supplies. USB supports plug and play attachment. When a new device is plugged into the bus, the change in power levels is detected and an address is assigned to the device which is unique on the bus. USB avoids contention by operating a master-slave protocol. USB uses NRZI encoding with bit-stuffing and CRC error detection.

Sometimes, as with Ethernet over twisted pair cables, the external interface from the computer is a serial synchronous interface direct from the NIC to an Ethernet hub, which acts as a repeater, in this case over a length of twisted pair telephone cable.

Activity 8.1

Study the V.24/RS232 interface in a textbook[2] or on a web site and examine the mechanical (connector type), electrical (voltage levels), functional (pin assignments) and procedural (order of signals exchanged) aspects of the specification.

[2] *For example, Stallings: pages 194–201.*

8.3 Functions

8.3.1 Basic physics of transmission

8.3.1.1 Sine waves

Many functions of the physical layer are dependent on the physical medium and its characteristics. Each physical medium therefore requires its own physical layer. Before we look at the functions of the physical layer, we should revise some basic physics and consider the different types of physical media. In order to understand the physical layer you must understand the basic physics of electromagnetism including electric conduction and electromagnetic radiation.

An important concept that occurs in many different communication systems is that of the sine wave. Light and other electromagnetic radiation, as well as alternating current electricity, all are examples of sine waves. A sine wave is a periodic wave that oscillates regularly and smoothly between a positive and a negative value. The time interval between one oscillation and another (when the wave exactly repeats itself) is known as the **period** (T) of the wave. The inverse of the period is known as the **frequency** (f) of the wave and the distance between one oscillation and the next is known as the **wavelength** (λ). The maximum height of the wave (at its crest) is known as the **amplitude** (A). Finally, the **phase** (ϕ) of the wave is a measure of how the wave is displaced in space or time. Two sine waves which can be superimposed exactly on each other are said to be in the same phase, but if one of the waves of the same frequency has its crests slightly later than the other so that they cannot be superimposed on each other, then the two waves are said to be out of phase. The three main characteristics of sine waves are illustrated in Figures 8.1 to 8.3 below.

Figure 8.1: Sine Wave Characteristics – Amplitude

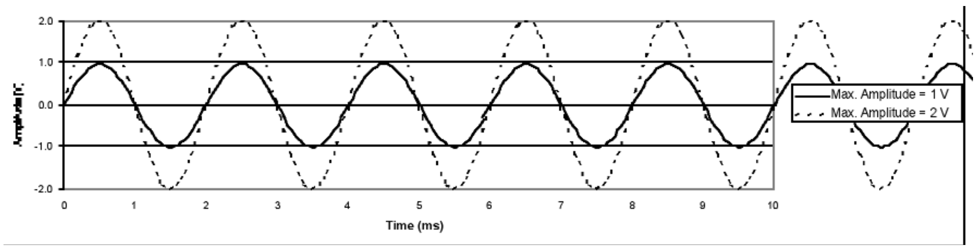


Figure 8.2: Sine Wave Characteristics – Frequency

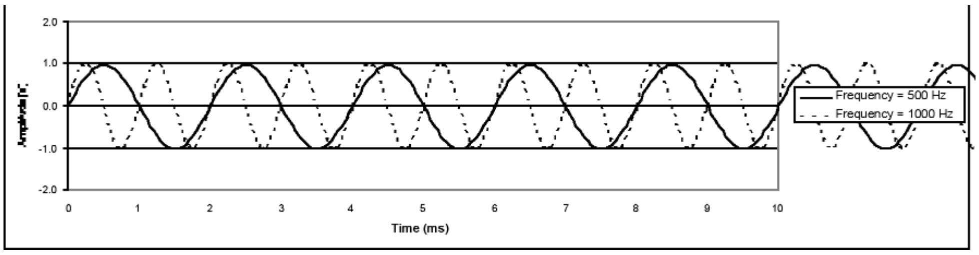
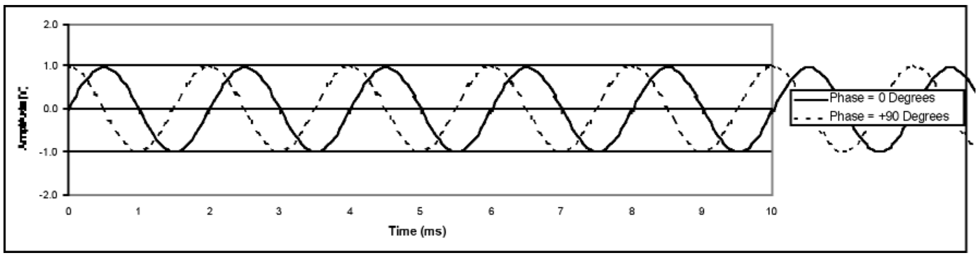


Figure 8.3: Sine Wave Characteristics – Phase



These characteristics of sine waves are related to each other by the following formulae:

$f = 1/T$

$c = f\lambda$

where c is the speed at which the wave propagates. For electromagnetic waves in free space, this is the speed of light which is roughly 3×10^8 m/s. In glass fibre light only travels at about 2×10^8 m/s, which is also roughly the same speed as electrical signals travel through a metallic circuit.

Activity 8.2

Before proceeding, you should study or revise your knowledge of the electromagnetic spectrum, in a textbook[3] or from a web site. A summary table describing the electromagnetic spectrum can be found in Table 8.1 below.

[3] *For example, Tanenbaum, pages 100–101.*

Table: 8.1: The Electromagnetic Spectrum

Radio Band	Frequency Ranges				Wavelength Ranges				Description / Communications Use
	Min	Max	Min (Hz)	Max (Hz)	Min	Max	Min (m)	Max (m)	
	0 Hz	3 Hz	0	$3x10^0$	100 Mm	∞	10^8	∞	
	3 Hz	30 Hz	$3x10^0$	$3x10^1$	10 Mm	100 Mm	10^7	10^8	
ELF	30 Hz	300 Hz	$3x10^1$	$3x10^2$	1 Mm	10 Mm	10^6	10^7	Extremely Low Frequency
ILF	300 Hz	3 kHz	$3x10^2$	$3x10^3$	100 km	1 Mm	10^5	10^6	Intra Low Frequency
VLF	3 kHz	30 kHz	$3x10^3$	$3x10^4$	10 km	100 km	10^4	10^5	GPS, Navigation beacons
LF	30 kHz	300 kHz	$3x10^4$	$3x10^5$	1 km	10 km	10^3	10^4	LW & Maritime Radio (AM)
MF	300 kHz	3 MHz	$3x10^5$	$3x10^6$	100 m	1 km	10^2	10^3	MW Radio (AM)
HF	3 MHz	30 MHz	$3x10^6$	$3x10^7$	10 m	100 m	10^1	10^2	SW Radio (AM)
VHF	30 MHz	300 MHz	$3x10^7$	$3x10^8$	1 m	10 m	10^0	10^1	VHF Radio (FM)
UHF	300 MHz	3 GHz	$3x10^8$	$3x10^9$	100 mm	1 m	10^{-1}	10^0	Terrestrial TV, Mob. Phones
SHF	3 GHz	30 GHz	$3x10^9$	$3x10^{10}$	10 mm	100 mm	10^{-2}	10^{-1}	Satellite TV, Microwaves
EHF	30 GHz	300 GHz	$3x10^{10}$	$3x10^{11}$	1 mm	10 mm	10^{-3}	10^{-2}	Radar, Microwaves
THF	300 GHz	3 THz	$3x10^{11}$	$3x10^{12}$	100 μm	1 mm	10^{-4}	10^{-3}	Far Infra-Red
	3 THz	30 THz	$3x10^{12}$	$3x10^{13}$	10 μm	100 μm	10^{-5}	10^{-4}	Infra-Red
	30 THz	300 THz	$3x10^{13}$	$3x10^{14}$	1 μm	10 μm	10^{-6}	10^{-5}	IR, light and UV, fibre optics
	300 THz	3 PHz	$3x10^{14}$	$3x10^{15}$	100 nm	1 μm	10^{-5}	10^{-6}	Ultra-Violet
	3 PHz	30 PHz	$3x10^{15}$	$3x10^{16}$	10 nm	100 nm	10^{-6}	10^{-7}	Extreme Ultra-Violet
	30 PHz	300 PHz	$3x10^{16}$	$3x10^{17}$	1 nm	10nm	10^{-7}	10^{-8}	X-Rays
	300 PHz	3 EHz	$3x10^{17}$	$3x10^{18}$	100 pm	1 nm	10^{-8}	10^{-9}	X-Rays
	3 EHz	30 EHz	$3x10^{18}$	$3x10^{19}$	10 pm	100 pm	10^{-9}	10^{-10}	X-Rays
	30 EHz	300 EHz	$3x10^{19}$	$3x10^{20}$	1 pm	10 pm	10^{-10}	10^{-11}	Gamma Rays

The amplitude of the sine wave a(t) at time t is given by the general formula:

$a(t) = A \sin (2\pi ft + \phi)$

Because the argument to the sine function in mathematics is an angle, the displacement of the wave in space and time is represented by a phase angle rather than by distance or time.

Activity 8.3

Using the formula, $a(t) = A \sin (2\pi ft + \phi)$, model the sine function in a spreadsheet with A (in Volts), f (in Hertz) and ϕ (in degrees) as parameters defined in separate cells at the top of the spreadsheet and t as a variable in each subsequent row ranging from 0 to 10 milliseconds (ms) in steps of 0.1 ms with a(t) calculated in the adjacent cell in each row. Use the autofill feature to create the 101 values of t and to copy the formula for calculating a(t) in each row. Plot a(t) against t in an X-Y (Scatter) chart and experiment with different values of A, f and ϕ. You should be able to produce charts similar to Figures 8.1 to 8.3.

The importance of sine waves in data communications is not just that signals are often carried by sine waves (as in fibre optics, wireless transmission and in communications using modems), but also that the shape of all signals (including a square shaped digital signal) can be represented by a series of sine waves (possibly infinitely many) of different frequencies and that this representation helps engineers to analyse and design transmission systems. The frequencies used are multiples of a base frequency and are known as **harmonics**. The mathematics behind this is known as Fourier[4] Analysis and

[4] *Named after Joseph Fourier, a French mathematician.*

is beyond the scope of this course. We shall simply accept that a signal of any shape can be represented by series of sine waves of different harmonic frequencies. Signals can therefore be viewed either by the variation of amplitude with time (the time domain) or by the amplitudes of their component frequencies (the frequency domain). There are two extreme cases to consider. If the signal does not change at all over time, then the frequency range of the component sine waves is zero (the signal, if it can be regarded as a signal, requires no bandwidth whatsoever) and if the signal changes abruptly from one level to another, as it would in a square wave, then the frequency range of the component sine waves is infinite (the signal requires infinite bandwidth if it is to be faithfully reproduced at the receiver).

No physical medium is capable of supporting an infinite range of frequencies (in other words supporting infinite bandwidth) and so all media will have to remove the higher frequency components from the signal. No medium can therefore perfectly carry a square shaped wave. The shape will be distorted by the medium when the higher frequency components are removed and the shape of the signal received will not have perfectly straight edges.

Activity 8.4

Using the Fourier series that generates a square-shaped signal:

$a(t) = \sin(2\pi ft) + \sin(2\pi f3t)/3 + \sin(2\pi f5t)/5 + \sin(2\pi f7t)/7 + \ldots\ldots$

with f = 500 Hz, create a spreadsheet that calculates the value of a(t) with just the first term and then the first two terms and then the first three terms etc., up to the first ten terms in the series, with the same values of t between 0 and 10 ms, in steps of 0.1 ms, as used in Activity 8.3. Plot the results in X-Y (Scatter) charts. With just one term, the resulting graph will be a standard sine wave, which will be how a square-shaped signal will be received via a channel whose bandwidth is severely limited so that only the base frequency can be transmitted. As each successive term is included in the calculation, the effect of increasing the bandwidth to include a further harmonics means that the square-shaped signal will be more faithfully received.

8.3.1.2 Types of physical media

Having reviewed some of the limitations of real media and studied the electromagnetic spectrum, we will now review the main types of physical media that were introduced in a level one unit. Physical media, as we discovered in Chapter 1, can be categorised into two main types (guided and unguided).

Guided media

Guided media constrain the propagation of the signals into a solid cable (either metallic or glass) that has to be laid, which is expensive. Signals do not easily stray from the cable and so attenuation is relatively low, and hence energy requirements are low and they are relatively secure from eavesdropping. They generally provide good error performance.

Guided media include:

- **Unshielded Twisted Pair** (**UTP**) cables, otherwise known as telephone cables. With UTP pairs of metal cables are twisted around each other in a regular fashion. Each metal conductor is surrounded by a plastic sheath to insulate it from other conductors and several (usually six) twisted pairs are then surrounded by an outer plastic sheath for protection. UTP comes in different grades at different costs. 10 Mbit/s Ethernet requires Category 3 (Cat 3) UTP while 100 Mbit/s Ethernet requires Category 5 (Cat 5) UTP.

- **Shielded Twisted Pair** (**STP**) cables are similar to UTP cables except that the outer sheath contains a metal mesh which is connected to the earth. The effect of this is that the pairs inside the sheath are protected from electromagnetic interference which means that STP cables are less subject to impulse noise, and can hence carry signals at higher speeds or over longer distances than can UTP. STP is more expensive, heavier and harder to install than UTP but provides better performance.
- **Coaxial cables (Coax)**, also used for connecting TVs to aerials, consist of an inner metal core and an outer metallic mesh conductor to shield the core from electromagnetic interference. The core and shield are separated by a plastic insulator. Finally, there is an outer plastic sheath for protection. Coaxial cables are bulky, heavy, expensive and difficult to install, but they do provide excellent performance. Early 10 Mbit/s Ethernets used coaxial cables as did long distance telecommunications circuits carrying multiple analogue voice channels using FDM before the invention of digital fibre optics.
- **Fibre optic cables**, carry light rather than the electrical currents that all of the above cables carry. The fibres are of very narrow diameter glass (about the same diameter as a human hair) surrounded by a glass cladding and then a plastic jacket for protection. At the transmitter, an electrical signal is converted into a beam of light by means of a Light Emitting Diode (LED) or a semi-conductor LASER and is guided down the fibre by total internal reflection. When the beam of light hits the boundary of the fibre core and the glass cladding, which have very different refractive indices, virtually all of the light is reflected back to remain inside the fibre. At the receiver, a photodiode converts the light signal received into an electric current. The main advantage of optical fibre is that it is very cheap, immune from electromagnetic interference and can carry signals at much higher speed and over much longer distances than electrical conductors. There are three types of fibre used.
  - **Multimode step-index fibre**, described above, where there is a sudden change in the refractive index between the fibre core and the glass cladding, which allows the light to take many different paths as it is reflected down the fibre which tends to disperse the signal.
  - **Multimode graded-index fibre** where there is a more gradual change in the refractive index, which allows the light to follow a number of curved paths down the fibre with less dispersion.
  - **Monomode fibre** has a smaller diameter (equal to a few wavelengths of the light) and in this mode, together with a highly focussed and directional beam, the light can only take one path down the fibre, which greatly reduces dispersion.

  There are four transmission bands within the electromagnetic spectrum that have low attenuation and are thus suitable for optical communication. They are all in the infrared region and potentially each can support huge capacity. One important point to note about fibre optic communication is that the signals are actually analogue, in that they carry light which is a continuous sine wave, but they carry digital data in simple on-off pulses.

Unguided media

With unguided media there are no or few constraints on the propagation of the signals. At higher frequencies they tend to follow line of sight, but even here the signal strays. Unguided media suffer more attenuation, require more energy and are vulnerable to eavesdropping. They are more subject to interference and hence errors and often fail in certain atmospheric conditions.

Unguided media include:

- **Radio transmission** is by far the most common form of unguided media and makes use of a very large part of the electromagnetic spectrum, with different frequency ranges having very different methods of propagation. Generally speaking, the higher the frequency the more the radio signal behave like light and travels in a straight line between the transmitter and the receiver requiring line-of-sight between them.
  - **Surface waves** follow the surface of the earth and are the means by which VLF and LF signals are propagated.
  - **Space (or tropospheric) waves** generally follow line of sight paths, but also can be reflected off the ground and other large objects. Radio broadcasting uses this method of propagation in the MF, HF, VHF and UHF bands.
  - **Sky (or Ionospheric) waves** are reflected off the ionosphere (50–402 km above the surface of the earth) and this enables them to be carried over long distances making this method ideal for long distance radio communication in the MF and HF bands.
  - **Scattered waves** rely on scattering rather than reflection. A very high powered signal in the VHF or UHF band is broadcast and the troposphere scatters it in all directions. Some of the signal is scattered back to earth and can be picked up by receivers. This method is often used to communicate with off-shore oil platforms when they are below the horizon.
  - **Satellite microwave communications** also use radio waves in the microwave band, but the signals are transmitted to a satellite, often in geo-stationary orbit (36,786 km above the earth), so that it orbits at the same speed as the earth rotates and hence appears in the same position relative to the earth. Some satellite communications, particularly for mobile uses, involve multiple satellites in low earth orbit, which are not stationary relative to the earth, but one is always within range. Satellite communications must avoid areas of the spectrum where gases in the atmosphere absorb signals. The main disadvantage of satellite communications is the propagation delay in carrying a signal up to the satellite and back to earth. For geo-stationary orbit the round trip time is about 250 ms.
  - **Terrestrial microwave communications** use line-of-sight between two (usually parabolic) dishes mounted on top of buildings or masts on hills. Whole networks have been built using terrestrial microwave, as it is often cheaper to install than laying cables. Its main disadvantage is that its performance can be adversely affected by the weather. There are likely to be a few days every year when atmospheric conditions are such that terrestrial microwave transmission will be unusable (e.g. very heavy rain or snow storms). Even on good days the error performance on microwave systems is significantly worse than on fibre optic cables.
- **Infra-red transmission in free space** is often used between two close buildings within line of sight of each other. Its main advantage is that, apart from the cost of the transmitters and receivers there are no other ongoing costs. It is much cheaper than leasing a circuit from a network operator. Unlike radio communications, no licensing is necessary. Infrared

communications are also used for remote controllers for TV and videos, and are sometimes used for wireless communications within a room (e.g. an Infra-red Data Association (IrDA)) wireless connection between a keyboard and a PC. Infra-red transmission requires line of sight. It will not pass through solid objects and will not work well outside in certain weather conditions such as heavy rain, snow and fog.

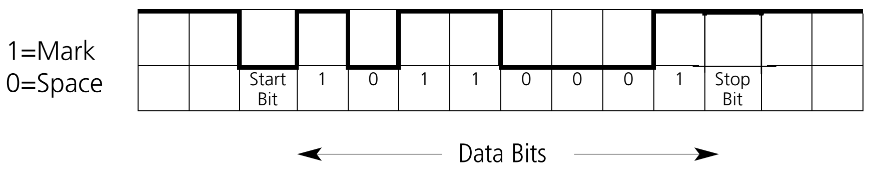
- **Laser transmission in free space** is similar in concept and has similar advantages and disadvantages to infra-red communications between buildings, also requiring line-of-sight, but uses coherent LASER light instead of infra-red signals.

8.3.2 Synchronisation

The physical layer simply transports bits, but at the receiver it must be able to determine when to sample each bit. There are two different techniques that can be used. They are **asynchronous** and **synchronous transmission**.

In Chapter 2, we considered asynchronous and synchronous communications channels. At the physical layer, these can be seen as the two different transmission methods. With asynchronous transmission, characters (or bytes) are sent one at a time down the physical link at any time, preceded by a start bit and followed by a stop bit, with the line being kept in an idle (mark or 1) state when no characters are being transmitted. The receiver is awoken from an idle state when it receives a start bit (space or 0). It must then sample the data at the speed of the line to receive 8 bits and then it receives the final stop bit (mark or 1), before returning to the idle state (mark or 1). Asynchronous transmission can be regarded as the transmission of a character as a very short frame with a start bit as a header and a stop bit as a trailer. Synchronisation therefore only occurs for the transmission of a single character, after which any further characters require re-synchronisation by the transmission of another start bit.

Figure 8.4: Asynchronous Transmission of 10110001



Synchronous transmission, on the other hand, involves the transmission of data in larger blocks, in fixed time slots with no start and stop bits surrounding each character. The blocks themselves contain headers and trailers which allow the receiver to synchronise with the transmitter to receive a whole block of data. Synchronous transmission often involves the transmission of a clock signal within the data signal, that allows the receiver to maintain synchronisation continuously. We shall examine how this can be achieved in Section 8.3.5.4.

8.3.3 Connection control

The physical layer provides a connection between two or more data link layer entities. It often supports a facility for the data link layer to activate and deactivate the physical layer connection. Ethernet is an exception in this regard, as it is a completely passive medium that requires no activation.

8.3.4 Multiplexing

At the physical layer, multiplexing allows the physical medium to be shared between two or more physical connections, which could be carrying the same or different data link layer protocols. The concept of multiplexing was introduced in Section 2.3.1. Multiplexing is a common method used at the physical layer to share capacity on physical links. The multiplexing technique used can be FDM or TDM or Statistical (also known as Asynchronous) TDM. With the latter two the physical layer will control a frame structure that will allow bits from each of the physical connections to be interleaved on the physical medium. The receiver is kept in synchronisation with the transmitter so that the correct bits from the frame are demultiplexed to recreate the original data that was multiplexed. The frame structure will include some header information which is regarded as a physical layer header.

With fibre optics, several signals can be carried at different wavelengths (or colours). This is a form of FDM, but in optics it is called **Wavelength Division Multiplexing** (**WDM**) and if the wavelengths of each signal are close to each other, it is called **Dense Wavelength Division Multiplexing** (**DWDM**).

8.3.5 Error control

The physical layer usually does not concern itself with the detection and correction of errors. These functions are the responsibility of higher layers. The physical layer is more concerned with the prevention of errors in the first place. Physical layer specifications will include details of the maximum bit rate that can be carried and the maximum distance over which signals can be propagated. This is influenced by the following transmission impairments, some of which were introduced in Chapter 2:

- **Attenuation** is the loss of signal strength over distance which can result in signal levels becoming so weak that they are not detected.
- **Noise** can come from many different sources. There are four basic types of noise: **thermal noise** which is caused by the random motion of electrons in a conductor (either a metallic communications circuit or within electronic communications equipment) as a result of heat. It is uniformly distributed across all frequencies and it is also known as **white noise** or **Gaussian noise**; **inter-modulation noise** which is caused when specific combinations of frequencies interfere with each other; **crosstalk noise** where signals from one communication channel are picked up on another channel by means of electromagnetic induction, as happens when two unshielded telephone cables are in close proximity; and **impulse noise** which, unlike the other types of noise, is short and intense, causing burst errors, and totally unpredictable. Hence it is difficult to design systems to overcome its effects. It is caused by such events as lightning, electric motors or lights being switched on and off and by vehicle ignition. With analogue voice signals, impulse noise will be evident as clicks and crackles, and whilst it is annoying, it will not significantly impair communications. With digital transmission, impulse noise will result in short bursts of bit errors, which will result in major problems if they are not detected and corrected.
- **Delay distortion** (also known as **inter-symbol interference**) is caused by media carrying different frequencies at slightly different speeds which over long distances will cause the shape of a signal to spread and one signal to begin to overlap another one.

- **Limited bandwidth**, which Fourier analysis has shown to be a constraint on all physical media and results in the shape of signals being distorted as the higher frequency components of digital signals are not transmitted. Over a long distance signals may become so distorted that they are unrecognisable.
- **Echoes** are caused when some of the energy of a signal is reflected back along a transmission medium. They usually occur at points where there is a change in the medium, such as its termination, or a point where 2-wire and 4-wire metallic circuits are connected together which always happens when a 2-wire local loop[5] is connected to a telephone switch.

[5] The metallic cable pair that runs from a local telephone exchange to a customer's premises, is often called the local loop. When the telephone handset is lifted, the two wires of the cable pair are connected together to form a loop which is directed by the exchange which then provides dial-tone.

The correct choice of physical layer can reduce most of these transmission impairments to manageable proportions. The choice of digital transmission and the use of repeaters at regular intervals to regenerate signals will reduce the effects of attenuation, noise, delay distortion and bandwidth limitations. Attenuation can be reduced by choosing a medium with low attenuation characteristics such as optical fibre. The effects of noise can be greatly reduced simply by shielding metallic cables or using media such as fibre optics that are not prone to electromagnetic interference. The effects of echoes are reduced on Ethernet LAN buses by terminating the bus with a high resistance. Other physical layers make use of totally separate channels for each direction (4-wire circuits) or echo suppression or echo cancellation techniques for 2-wire circuits such as the local loops that connect telephone subscribers to the PSTN.

- **Echo suppression** works by sensing which party is transmitting and blocking all communications in the opposite direction, essentially making the channel half-duplex, which makes voice communications difficult and most data communications impossible. This technique has been used in the PSTN, particularly with inter-continental and satellite circuits where the effects of echoes (about 250 ms with satellites) are very disconcerting. In order to transmit full-duplex data over these circuits, it is necessary to disable the echo suppressors, which can be done by the transmission of a 1200 Hz tone down the circuit.
- **Echo cancellation** is a more modern and sophisticated technique. It works by predicting the strength and timing of the echo, and subtracting this from signals travelling in the opposite direction. Echo suppressors are being replaced by echo cancellers. ISDN and ADSL both use echo cancellation techniques.

The maximum bit rate achievable is dependent on the ability of the receiver to discern the level of signals received without errors. This depends on the effects of the transmission impairments above, particularly noise. There are two formulae which are useful in calculating the maximum theoretical bit rate that can be achieved.

Nyquist's theorem[6] considers how a receiver samples an incoming signal in a noiseless channel. It does this by measuring a signalling level (a voltage in the case of an electrical signal) at regular intervals corresponding to the bits transmitted. If two levels are used (one to represent a 0 and the other to represent a 1), Nyquist's Theorem concludes that:

[6] Named after Harry Nyquist of Bell Labs, a Swedish American communications engineer.

$$C = 2B$$

where C is the maximum theoretical capacity in bit/s and B is the bandwidth in Hertz.

This however is a specific case of the more general theorem. The general theorem gives the result:

$C = 2B \log_2 V$

where C is the maximum theoretical capacity in bit/s and B is the bandwidth in Hertz and V is the number of signalling levels that are used to carry the signal.

If, for instance, a signal uses 8 different signalling levels then each level could be used to represent 3 (= $\log_2 8$) bits. So the first level could be used to represent 000, the second 001, etc.

Note that, if there are only two signalling levels, the above formula reduces to the earlier result C = 2B (as $\log_2 2 = 1$).

You may think that by continually increasing the number of signalling states, that it will be possible to achieve higher and higher bit rates, but this is not so, as in practice, there is no such thing as a noiseless channel and any noise in the channel will make it harder for the receiver to distinguish between the many signalling levels.

As an example, consider a 3.1 kHz telephone circuit in which a signal is carried using 16 signalling levels. Nyquist's Theorem gives the maximum theoretical capacity of the channel as:

$$\begin{aligned} C &= 2B \log_2 V \\ &= 2 \text{ x } 3{,}100 \log_2 32 \\ &= 6{,}200 \text{ x } 5 \\ &= 31{,}000 \text{ bit/s} \\ &= 31 \text{ kbit/s} \end{aligned}$$

Note: This example illustrates the difference between the bit rate, which is the quantity of data transmitterd per second, in this case 30,000 bits per second, and the **Baud rate**,[7] which is the number of signalling level changes or **symbols** per second. In this example the Baud rate would be 6,000 Baud.

Shannon's Law[8] considers a noisy channel and takes account of the signal to noise ratio in calculating the maximum theoretical capacity of the channel. Shannon's Law states that:

$C = B \log_2(1 + S/N)$

Again, we shall look at a simple example of a 3.1 kHz telephone circuit with a Signal to Noise Ratio of 40dB, which is above what can be normally expected. We must first convert 40dB to an real ratio.[9]

$$\begin{aligned} 40 &= 10 \log_{10} S/N \\ \log_{10} S/N &= 4 \\ S/N &= 10^4 \\ &= 10{,}000 \end{aligned}$$

Shannon's Law then gives:

$$\begin{aligned} C &= B \log_2(1 + S/N) \\ &= 3{,}100 \log_2(1 + 10{,}000) \\ &= 3{,}100 \log_2(10{,}001) \\ &= 3{,}100 \text{ x } 13.2879 \\ &= 41{,}192 \text{ bit/s} \\ &= 41.2 \text{ kbit/s} \end{aligned}$$

[7] Named after Maurice Emile Baud, a French pioneer of telegraphy.

[8] Named after Claude Shannon of Bell Labs, an American communications engineer and the father of Information Theory.

[9] See Section 2.3.2 of this volume.

8.3.6 Data encoding

Fourier analysis has shown that infinite bandwidth is required to faithfully transmit a square-shaped digital signal. As no physical medium can offer infinite bandwidth, the physical layer must filter out the higher frequency components. After doing this, the remaining frequencies may still not be suitable for transmission on the physical medium. If the physical medium is the PSTN, then the frequency range must be within the range 300-3400 Hz which is used to carry analogue voice signals. If the physical medium is optical fibre then the frequency range must be within the range for visible or ultra-violet light (185-366 THz). In order to be able to transmit data on these different physical media, it is necessary to superimpose the data to be carried on an analogue sine wave known as a **carrier signal**, with a frequency in the range required, known as the **carrier frequency**. This process of translating a data signal to a different frequency range is known as **modulation** and the reverse process is known as **demodulation** and a device that performs this function is known as a **modulator/ demodulator** or **modem**. If analogue data is transmitted without superimposing it on a carrier frequency (i.e. baseband transmission), the device is known as a **coder/decoder** or **codec**.

There are four different types of data transmission, which are described in more detail by Stallings.[10] We shall use the same classification, in discussing modulation and coding, although in a different order.

[10] *Stallings, Chapter 5, pages 130–166.*

8.3.6.1 Analogue data, analogue signals

Analogue data such as voice or video is modulated onto an analogue carrier signal. The carrier signal is varied according to the shape of the analogue data. Any of the three characteristics of the carrier sine wave can be varied:

- **Amplitude Modulation** (**AM**) is where the amplitude of the carrier signal is varied according to the analogue data being carried. If a continuous line is drawn through all the crests of the modulated carrier signal then the shape of this line will be the same as the analogue data being carried. AM is used by medium wave and long wave radio broadcasting.
- **Frequency Modulation** (**FM**) is where the frequency of the carrier signal is varied according to the analogue data being carried. The analogue data in FM is harder to visualise than it is in AM. FM is used by VHF radio broadcasting and was also used to separate voice signals in Frequency Division Multiplexing.
- **Phase Modulation** (**PM**) is where the phase of the carrier signal is varied according to the analogue data being carried. The analogue data in PM is even harder to visualise than it is in FM. PM is not often used to carry analogue data.

8.3.6.2 Analogue data, digital signals

The analogue data received from a telephone has to be digitised by a codec at the local exchange before it can be transmitted. The PSTN, which has a digital core network, must do this to carry analogue voice. The techniques it uses are:

- **Pulse Amplitude Modulation** (**PAM**) is used to sample the analogue sound waves. The PSTN transmits voice in a 3,100 Hz frequency range. Nyquist's theorem would indicate that the signal must be sampled at a rate of at least 6,200 Hz if the waveform is to be successfully reproduced with equivalent quality. In fact, PAM conservatively samples the analogue voice signal every 125 μs and therefore produces 8,000 pulses per second that characterise the analogue signal.

- **Pulse Code Modulation** takes the pulses obtained by PAM and digitises them by a process known as **quantisation**. It measures the amplitude of pulses and assigns them to the nearest of 256 quantised levels. Each level is then given an 8-bit code to represent the amplitude of the pulse. From this process a stream of binary data is produced. It will consist of 8,000 samples every second each coded as 8 bits giving a total of 64 kbit/s which is the standard rate for a digital voice channel. The PCM quantisation process results in small errors when the signal is reconstructed. This is known as **quantisation noise**. It can be reduced by choosing quantised levels that are not equally spaced with the levels being more closely spaced for low amplitudes than for high amplitudes. This process is known as **compression/expanding** or companding. There are two companding algorithms used: μ-law in North America and Japan and A-law which is the ITU-T method used in the rest of the world.

Once the voice signals are encoded in PCM, they can be transmitted digitally. At the far end, the last telephone switch has a codec that converts the PCM back to PAM and transmits this down the analogue phone line to a telephone. The quality of the signal received at this phone is not noticeably different to that received if the phone call was carried by analogue transmission, but is significantly better in the presence of noise.

Note that PCM does not use bandwidth efficiently. Modern compression algorithms can transmit voice at rates as low as 6 kbit/s, but 64 kbit/s is the standard for transmitting voice, and forms the basic unit for all subsequent multiplexing on carriers' transmission networks.

8.3.6.3 Digital data, analogue signals

The digital data has to be modulated onto an analogue carrier signal. This is really a special case of modulating analogue data onto an analogue carrier signal and the same techniques are used, but they are now called shift keying, as the changes in the data are abrupt. Similar to analogue data modulation, the three basic types are:

- **Amplitude Shift Keying** (**ASK**) is where the amplitude of the carrier signal is varied according to the digital data being carried. The amplitude of the wave shifts between two values (one is often 0). If a continuous line is drawn through all the crests of the modulated carrier signal, then the shape of this line will be the digital data signal being carried. ASK is inefficient in its use of bandwidth and sensitive to interference when used over metallic circuits. It is not used on its own with modems, but it is used to modulate digital data onto an optical fibre.
- **Frequency Shift Keying** (**FSK**) is where the frequency of the carrier signal is varied according to the digital data being carried. FSK uses a number of different frequencies (or tones if in the voice frequency range) and shifts between them. For full duplex transmission, different frequencies will be used to carry data in each direction. FSK is more efficient in its use of bandwidth and less prone to errors than ASK. It is also used on low-speed modems (up to 1,200 bit/s).
- **Phase Shift Keying** (**PSK**) is where the phase of the carrier signal is varied according to the digital data being carried. PSK uses a number of different phases to represent the digital data being transmitted and shifts between them. PSK is less prone to interference than ASK and FSK and can therefore work at higher speeds (up to 4800 bit/s). Four phases are used in 4-PSK and eight phases in 8-PSK. Each phase represents a symbol of 2 or 3 bits, respectively. PSK can be represented graphically in a **constellation diagram**, where each phase angle used is represented by the angle of a line through the origin of a graph.

Figure 8.5: Constellation Diagram for 2-PSK

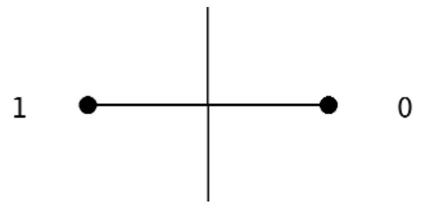
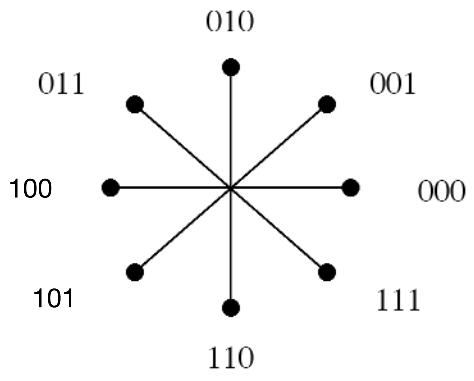


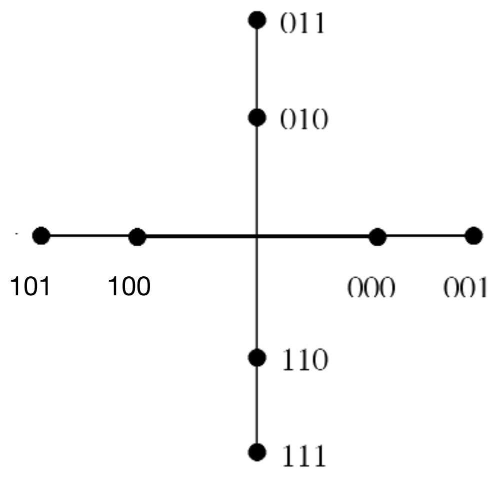
Figure 8.6: Constellation Diagram for 8-PSK



All of these methods have their limitations, but they can be used in combination to achieve much higher data rates.

- **Quadrature Amplitude Modulation** (**QAM**) is a composite of ASK and PSK. As an example, 8-QAM uses two amplitudes and four phases. QAM can also be represented in a constellation diagram, by using the lengths of the lines to represent amplitudes.

Figure 8.7: Constellation Diagram for 8-QAM



Higher bit rates can be achieved with more complex QAM constellations and even higher rates can be achieved by using a variation of QAM called **Trellis Code Modulation** (**TCM**), so named because the constellation diagram looks like a trellis. This uses extra parity bits per symbol for Forward Error Correction. Using trellis coding, data rates of up to 33.6 kbit/s can be achieved.

8.3.6.4 Digital Data, Digital Signals

Digital data can be transmitted digitally without any alteration to the data whatsoever, but this is rarely done. There are some good reasons why it may be necessary to code the data differently. These include:

- Shaping the frequency spectrum to:
  - limit high frequency components and thus reduce bandwidth and improve efficiency
  - avoid Direct Current (DC) components (i.e. 0 Hz) in electrical media, as some electrical equipment such as amplifiers cannot handle DC components
  - concentrate transmitted power in the middle of the transmission bandwidth, as transmission characteristics are usually worse at the edges of a band.
- Reducing errors by means of rudimentary error detection.
- Maintaining synchronisation, by transmitting timing information along with the signal or by avoiding long runs at the same level.
- Providing security by means of data scrambling.

For these reasons digital data is nearly always further encoded before it is transmitted. The code that is used on the physical medium is called the **line code**. Many line codes were designed for use with electrical circuits where the signalling levels are voltages that can be positive or negative, but some of the line codes can also be used on optical fibres or radio systems.

Line codes are classified as follows by Forouzan.[11]

[11] *Forouzan, pp.89–95.*

- **Unipolar** uses two signalling levels of the same polarity (either positive or negative) or one signalling level and the zero level, with lower level representing 0 and higher level representing 1.
- **Polar** uses two signalling levels of opposite polarity.
  - **Non Return to Zero** (**NRZ**) – always positive or negative, never zero.
    - **Non Return to Zero – Level** (**NRZ-L**) – negative represents 0 and positive represents 1.
    - **Non Return to Zero – Invert** (**NRZ-I**) – no inversion at start of bit represents 0 and inversion at start of bit represents 1.
  - **Return to Zero** (**RZ**) – negative represents 0 and positive represents 1, but returns to zero in middle of bit.
  - **Biphase** uses two signalling levels of opposite polarity but inserts an extra transition in the middle of each bit to provide bit synchronisation, and as a result uses twice as much bandwidth, so it is only used in places where bandwidth is cheap, such as LANs.
    - **Manchester Phase Encoding**[12] (**MPE**) – mid-bit transition from negative to positive represents 0 and mid-bit transition from positive to negative represents 1.
    - **Differential Manchester Encoding**[13] (**DME**) – inversion at start of bit represents 0 and no inversion at start of bit represents 1.
- **Bipolar** uses three signalling levels, positive, zero and negative.
  - **Alternate Mark Inversion** (**AMI**) – zero level represents 0 and alternating positive and negative levels represent 1, but long strings of 0s can cause loss of synchronisation.

[12] *Used in 10 Mbit/s Ethernet LANs.*

[13] *Used in Token Ring LANs.*

  - **Pseudoternary**[14] – opposite of AMI with alternating positive or negative levels representing 0 and zero level representing 1.
  - **Bipolar n Zero Substitution**[15] **(BnZS)** – modified AMI where n consecutive 0s are substituted by a code with positive and negative levels which break the AMI rules by not alternating (code violation) so that receiver can re-substitute a string of n consecutive 0s.
  - **High Density Bipolar Order 3**[16] **(HDB3)** – modified AMI where four consecutive zeros are substituted by codes containing positive or negative levels with code violations.
  - **Code Mark Inversion**[17] **(CMI)** – a mid bit transition represents 0 and alternating positive and negative levels represent 1.
- **Other**
  - **2 Binary / 1 Quaternary**[18] **(2B/1Q)** – 2B means that two bits are coded and 1Q that they are coded as one symbol which has four different levels.
  - **Multi-Line Transmission 3-level (MLT-3)**[19] – similar to NRZ-I except it uses three levels. No transition at the start of a bit represents a 0 and a transition between two of the three levels represents a 1. The transitions cycle from 0 to positive to zero to negative to zero.
  - **Block Codes** – represent a string of bits by a code consisting of a longer string. The codes are chosen so that there are enough transitions to maintain synchronisation and the redundant bits can be used for error detection.
    - **4B/5B**[20] – a 4-bit binary string is coded as a 5-bit binary string.
    - **8B/10B**[21] – an 8-bit binary string is coded as a 10-bit binary string.
    - **8B/6T**[22] – an 8-bit binary string is coded as 6-symbol ternary (3-level) code.

[14] *Used in Basic rate ISDN lines.*

[15] *B8ZS is used in N. American T1 (1.5 Mbit/s) and B3ZS on T3 (45 Mbit/s) private circuits.*

[16] *Used in European E1 (2Mbit/s), E2 (8 Mbit/s) and E3 (34 Mbits/s) private circuits.*

[17] *Used in European E4 (140 Mbit/s) private circuits.*

[18] *Used in N. American Basic rate ISDN lines.*

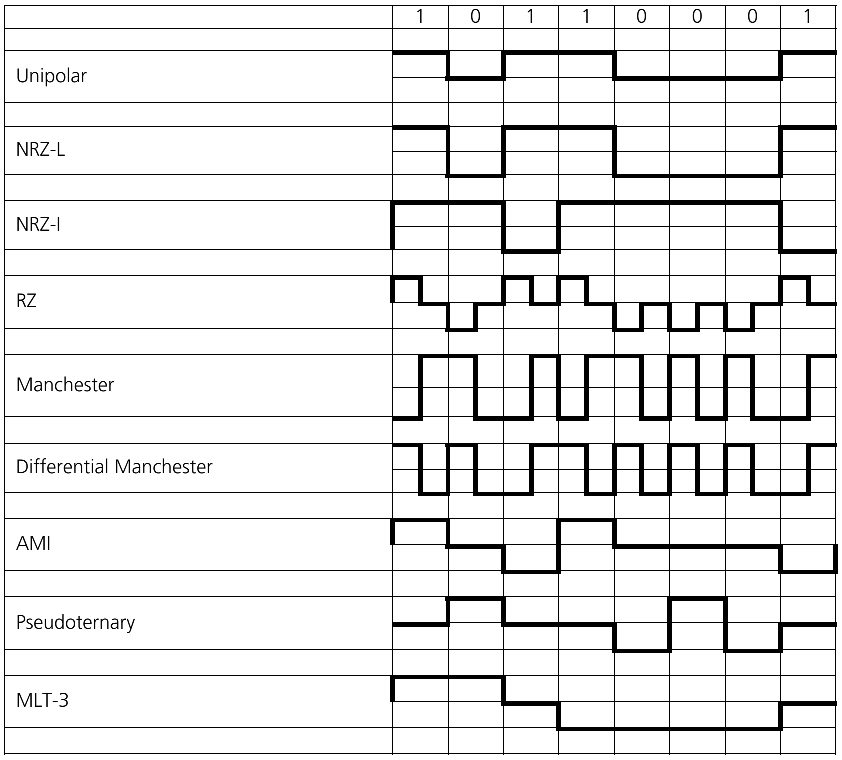
[19] *Used in 100Base-TX Ethernets.*

[20] *Used in 100Base-FX and 100BaseTX Ethernets.*

[21] *Used in Gigabit Ethernets.*

[22] *Used in 100Base-T4 Ethernets.*

Figure 8.8: Examples of coding 10110001 using different line codes



8.4 Dial-up modems

The ITU-T publishes modem standards in its V series recommendations. These cover the physical aspects of the modem, but do not define how the modem is controlled. This is done by in-band signalling using a set of ASCII commands to perform actions such as dialling and hanging up. Most modems use proprietary extensions of a *de facto* standard known as the Hayes standard, named after the company that developed it. It is also known as the AT standard, as all commands are preceded by AT (for attention), once command mode is entered by means of a +++ escape sequence. In reality, because of all the proprietary extensions, it is not a standard at all.

Table 8.2 lists the most important dial-up modem standards and illustrates the improvement in performance over time as superior modulation techniques were developed.

Table 8.2: ITU-T V Series Dial-up Modem Standards

Date	Standard	Async/Sync	Modulation Technique	Data Rate (bits/s)	Signalling Rate (Baud)
1964	V.21	Async	FSK	300	300
1964	V.23 (downstream) (upstream)	Async	FSK	1,200 75	1,200 75
1972	V.26 bis [23]	Sync	4-PSK	2,400	1,200
1976	V.27 ter [24]	Sync	8-PSK	4,800	1.600
1980	V.22	Async/Sync	4-PSK	1,200	600
1984	V.26 ter	Async/Sync	4-PSK	2,400	1,200
1984	V.22 bis	Sync	16-QAM	2,400	600
1984	V.32	Sync	32-TCM	9.600	1,920
1991	V.32 bis	Sync	128-TCM	14,400	2,400
1993	V.32 ter	Sync	512-TCM	19,200	2,400
1994	V.34	Sync	960-TCM	28,800	3,429
1995	V.34 bis	Sync	1664-TCM	33,600	3,429
1998	V.90 (downstream) (upstream)	Sync	PCM 1664-TCM	56,000 33,600	8,000 3,429
2000	V.92 (downstream) (upstream)	Sync	PCM PCM	56,000 48,000	8,000 8,000

[23] *Bis is the Latin word for twice. It is used for the second version of an ITU-T recommendation.*

[24] *Ter is the Latin word meaning three times (or thrice). It is used for the third version of an ITU-T recommendation.*

You do not need to learn the details of the above table, apart from the last four which are still in common use, but you do need to appreciate the way that data rates have steadily increased over the last forty years, as new modulation techniques have been developed. The earliest modems used FSK which was soon replaced by PSK. V.22 was the first ITU-T modem to implement a data compression algorithm. V.27 ter was the first standard to use echo cancellation techniques, V.22 bis the first standard to use QAM and V.32 the first standard to use TCM and its Forward Error Correction.

The V.34 bis data rate of 33.6 kbit/s is approaching the theoretical Shannon limit of 41.2 kbit/s that we calculated for an excellent SNR of 40 dB in Section 8.3.3. V.34 bis modems have to fall back to lower data rates if the SNR of the circuit is below 33 dB, as it often is.

Given the Shannon limitation, you may wonder how V.90 modems can possibly achieve 56 kbit/s. To achieve this, the bandwidth limitation of 3,100 Hz or the SNR must be improved. V.90 modems actually do both. They make use of the fact that ISPs connect to the PSTN via digital circuits that have very good SNRs. ISPs can transmit 56 kbit/s (in North America), or 64 kbit/s of digital data (elsewhere), down these circuits as far as the last telephone switch that supports the customer's phone line. From there the PCM codec converts the PCM to PAM for transmission to the V.90 modem. The V.90 data rate is asymmetric, 33.3 kbit/s is offered upstream which is sampled by the PCM equipment at the telephone switch and is within the Shannon limit for a good circuit. In the download direction, no sampling is required as the data is already digital and hence there is no quantisation noise. All of this means that 56 kbit/s is possible in the downstream direction, although it still requires a good quality local loop. The reason that V.90 does not offer 64 kbit/s is that it would not be possible in North America, as some bits are used for signalling. The V.90 standard was designed so that it could be used throughout the world.

V.92 is a new standard that offers a higher upstream data rate (up to 48 kbit/s) and also supports a call-waiting or on-hold facility that allows a voice call to be answered with the data call on-hold. Modem technology has now reached its limit for the PSTN, and improvements in data rates are now only achievable by bypassing the PSTN altogether, as happens with ADSL.

8.5 Digital Subscriber Line (DSL)

Before we consider DSL, we should familiarise ourselves with the topology of and operation of local telephone networks. Each telephone is connected via a local loop cable pair to the local telephone exchange[25] in a star network. There is no multiplexing within the local network. Each subscriber has its own dedicated cable pair and termination on the telephone switch. The 3.1 kHz limitation on the PSTN is not due to the local loop. It is there because the telephone switch limits the bandwidth, so that telephone calls can be multiplexed efficiently. A typical local loop can support a frequency range of about 1 MHz. DSL releases the potential of the local loop by not connecting it directly to the telephone switch.

[25] *In some countries, particularly in N. America, a local telephone exchange is called an end office or a local central office.*

Asymmetric Digital Subscriber Line (ADSL) is one of a family of broadband services that are being offered over local loops that bypass the PSTN. ADSL has been designed to provide high bandwidth access to the Internet or other services without having to re-engineer local loop circuits and without affecting normal telephony over the loops. ADSL is an asymmetric protocol that typically provides up to 1.5 Mbit/s downstream, (more typically 512 kbit/s or 1 Mbit/s) and up to 512 kbit/s upstream (more typically 256 kbit/s). It uses **Discrete Multi-Tone** (**DMT**) which splits the line into 256 downstream FDM sub-channels and 32 upstream, each 4.3125 kHz wide and capable of 4 kbaud signalling rates. Each symbol transmitted can convey between 2 and 15 bits. DMT adapts the number of signalling levels in each sub-channel according to the measured quality of the sub-channel. In theory, if every sub-channel could carry 15 bits, DMT would be able to achieve 15 Mbit/s, but real circuits are nothing like that good. Normal analogue voice telephony occupies sub-channel 0, and sub-channels 1 to 5 are not used to avoid interference between voice and data. The remaining sub-channels are used for data. A QAM modulation scheme, similar to V.34, is operated within each sub-channel. The quality of the line is constantly monitored and the data rate is adjusted in each sub-channel, if necessary.

Customers can connect their PCs to an ADSL modem via a 10 Mbit/s Ethernet port or a USB port and then plug the modem into a telephone socket. At the telephone exchange, the circuit has to be connected to a **Digital Subscriber Line Access Multiplexer** (**DSLAM**) instead of the telephone switch. The DSLAM filters out sub-channel 0 and connects it to the PSTN switch. The remaining upstream sub-channels are multiplexed into a single data stream, which is forwarded to the ISP via an ATM virtual channel.

In the opposite direction, an ATM virtual channel is inverse multiplexed into separate ADSL downstream sub-channels and these are frequency division multiplexed together with the voice sub-channel onto the ADSL line.

Activity 8.5

Using textbooks or web searches, find information about other DSL protocols such as VDSL and SHDSL.

8.6 Cable modems

Cable modems offer an alternative broadband service to ADSL in areas which have been cabled by cable TV companies. Again we need to understand the topology and operation of cable TV networks, before examining how cable modems work. Cable TV networks were originally designed to frequency division multiplexed 6 MHz analogue broadcast TV channels from a central site, called the head end, to individual homes via a cascaded star network with amplifiers at various points. Older cable TV networks were connected entirely with coaxial cables. There are flexibility points in the cascaded star network, usually in street cabinets where the cable branches. This type of network is unsuitable for carrying interactive data services, as there is insufficient spare capacity to support them, particularly on the cables closest to the head end. Newer cable TV networks and older ones that have been upgraded use a different approach. In these networks, the head end is connected to the street cabinets via fibre optic cable using digital transmission and this is then branched out to individual customers via coaxial cables using analogue transmission. Replacing the coaxial cables to the homes would be prohibitively expensive. This hybrid type of cable TV network is known as a **Hybrid Fibre-Coaxial** (**HFC**) network. These types of networks do have sufficient capacity to carry interactive data services to the head end, but they need to multiplex the data to and from each house onto a high bandwidth fibre channel.

The standard for cable modems is the **Data over Cable Service Interface Specification** (**DOCSIS**). The interface with the PC is normally a 10 Mbit/s Ethernet interface, but can also be a USB interface. 37 MHz of Bandwidth on the coaxial cable below the frequencies of the analogue TV channels is reserved for upstream data communications and 200 MHz is reserved at the higher frequencies above the analogue TV channels for downstream communications. Both of these bandwidth ranges are divided into 6 MHz channels (the same bandwidth as the TV channels). Like ADSL, cable modems provide asymmetric data rates. Downstream, data is modulated using 64-QAM with 6 bits for each baud. Of these, 5 bits are used for carrying data, and 1 bit for forward error correction. In theory, a cable modem can support 30 Mbit/s of downstream data, but the Ethernet interface will limit this to 10 Mbit/s. In the upstream direction, the lower frequency channels are more prone to noise and interference, and QAM is not suitable. Instead, the upstream data is modulated using 4-PSK with 2 bits for each baud. The maximum theoretical upstream bandwidth is therefore 12 Mbit/s, but in practice it is usually much less.

The 6 MHz channels in both directions support a fixed number of cable modems. Because the coaxial part of the network is a broadcast tree network, there is a multiple access contention problem to be solved. In the downstream direction, there is no contention problem, as there is only one transmitter (the head end) and it broadcasts data down the correct channel and only the addressed device will read the data. In the upstream direction, a minislot reservation access method is used on each channel to allocate bandwidth between contending devices. Before a cable modem can send data, it must be allocated to one of the upstream and one of the downstream channels. It must then determine its distance from the head end, so that it can send its reservation minislot at exactly the correct time. It does this by a process called **ranging** which involves sending a special packet to the head end and timing how long it takes for the response to be returned.

Specimen examination question

a. State whether each of the following statements is true or false and, if false, correct the statement:
   i. V.90 modems use Trellis Code Modulation for both upstream and downstream communications.
   ii. The transmission of a square-shaped signal in such a way that it is received as a square-shaped signal will require infinite bandwidth.
   iii. Higher data rates can be achieved with multimode fibre than with monomode fibre.
   iv. Pulse Code Modulation samples voice signals as using 7 bits at 8000 Hz.
b. Describe the main differences between guided and unguided media, giving three examples of each.
c. How many bits are represented by one symbol in 16-QAM modulation? What bandwidth will be required if 2,400 bit/s is to be transmitted by a 16-QAM modem.
d. Give two reasons why LANs tend to use biphase line codes such as Manchester and Differential Manchester.
e. Describe the main differences between ADSL and cable modems in providing broadband services to homes.

Learning outcomes

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

- list three common physical interface standards
- define the characteristics of a sine wave and how frequency and wavelength are related to each other
- appreciate that when square-shaped signals are transmitted, the received signal will be distorted by the absence of high frequency harmonics as a result of bandwidth limitations
- describe the different types of physical media, their properties and their use in communications
- describe the transmission impairments that can affect transmission at the physical layer and some of the methods that can be used to overcome them
- perform calculations on capacity and bandwidth using Nyquist's Theorem and Shannon's Law
- describe the different analogue modulation techniques and how they are related to dial-up modem standards
- outline how Pulse Code Modulation works
- explain the reasons for using line codes. Describe and classify four line codes in common use.

Appendix A

Specimen examination paper

Answer two questions from this section.

Question 1

a. State whether each of the following statements is true or false and, if false, correct the statement:

   i. The bit rate of a channel is always the same as the baud rate.

   ii. Differential Manchester encoding is used in Ethernets.

   iii. A combination of Longitudinal and Vertical Redundancy checks can correct a single bit error.

   iv. The Ethernet backoff algorithm can result in a frame not being transmitted. [3]

b. State Shannon's Law.

   What is the maximum capacity of a channel in bit/s, if the bandwidth of the channel is 3 kHz and the Signal to Noise Ratio is 30 dB? [6]

c. What type of noise is responsible for burst errors on communications circuits. What causes this noise? Suggest three different ways in which the effects of it can be reduced. [6]

d. Show how the 3-bit CRC is calculated to protect a 5-bit code 10110 using the generator 1001. [7]

e. Describe three different ways by which a start of a data link frame can be indicated. [3]

Question 2

a. State whether each of the following statements is true or false and, if false, correct the statement:

   i. The IPv4 interface only has two primitives (SEND and DELIVER).

   ii. IPv6 incorporates the functions of IPsec, IGMP, ARP and RARP.

   iii. TCP and UDP applications are identified by port numbers that are 8 bits long.

   iv. The TCP and UDP header checksums also protect a part of the IP datagram. [3]

b. Describe the protocol interactions that take place when a ping command is executed. [6]

c. Describe the two modes of operation used by IPsec. [6]

d. Briefly describe four important functions of the transport layer. [4]

e. Describe how a TCP connection is identified and how this identification is unique. [6]

Question 3

a. State whether each of the following statements is true or false and, if false, correct the statement:

   i. MIME data types are used by many Internet applications, not just email.

   ii. The HTTPS protocol is identical to HTTP but uses TLS as another transport layer protocol above TCP.

   iii. Transmission delay is directly proportional to distance.

   iv. An application layer client server protocol is normally implemented as a symmetric connectionless protocol. [3]

b. With the aid of a diagram, describe Shannon's communications model, showing all the components that are affected by noise. [5]

c. Distinguish between *de jure* and *de facto* standards and give an example of each. [5]

d. Identify three reasons why an applications developer might choose to use an unreliable transport service. [6]

e. Describe how password security is implemented in HTTP. [6]

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

<u>00100100 11100100</u>

<u>10001000 01110001</u> = checksum

10101100 10011001 (errors in bits 3 and 4)

00100100 11100100

<u>10001000 01110001</u>

<u>00000000 00001100</u>

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

Its network address is therefore 192.16.128.0/13 and the netmask is 255.255.248.0.

d. See Section 6.4.4.

e. See Section 6.8.

Chapter 7

a. True or False

   i. FALSE – they are 24 bits long.

   ii. FALSE – it's CSMA/CD.

   iii. TRUE.

   iv. TRUE.

b. See Section 7.3.6.2.

c.

	b12	**b11**	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**
	1	0	1	0		1	0	0		1		
b1		0		0		1		0		1		**0**
b2		0	1			1	0			1	**1**	
b4	1					1	0	0	**0**			
b8	1	0	1	0	**0**							

Even Hamming code = 101001000110.

	b12	**b11**	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**fix**
	1	1	0	0	0	1	1	0	0	1	0	1	
b1		1		0		1		0		1		**1**	0
b2		1	0			1	1			1	**0**		0
b4	1					1	1	0	**0**				1
b8	1	1	0	0	**0**								0

The error is in bit 4 (a parity bit), so no bits in the original byte were corrupted.

The byte transmitted was 11001101.

d. See the second bullet point in Section 7.6.

e. See the second paragraph in Section 7.8.

Chapter 8

a. True or False

   i. FALSE – it uses QAM upstream.

   ii. TRUE.

   iii. FALSE – higher data rates are achieved with monomode.

   iv. TRUE.

b. See Section 8.3.1.2.

c. Using Nyquist's Theorem

$C = 2{,}400$ bit/s

$V = 16$

$C = 2B \log_2 V$

$2{,}400 = 2B \times \log_2 16$

$2{,}400 = 2B \times 4$

$8B = 2{,}400$

$B = 300$ Hz

d. See Section 8.3.6.4

e. See Sections 8.5 and 8.6.

A: Specimen examination paper

Question 1

a. True or False

   i. FALSE – only the same if there are two signalling levels.

   ii. FALSE – it's Manchester encoding.

   iii. TRUE.

   iv. TRUE.

b. $C = B \log_2(1 + S/N)$

   where C is capacity in bits/s

   B is bandwidth in Hz

   S/N is Signal to Noise Ratio

30	=	$10 \log_{10} S/N$
$\log_{10} S/N$	=	3
S/N	=	10^3
	=	1,000
B	=	3,000 Hz
C	=	$B \log_2(1 + S/N)$
C	=	$3{,}000 \log_2(1 + 1{,}000)$
	=	$3{,}000 \log_2 1{,}001$ bits/s

c. See Section 8.3.5.

d.

```
                     1 0 1 0 0
1 0 0 1 ) 1 0 1 1 0 0 0 0
        ⊕ 1 0 0 1
            0 1 0 0
          ⊕ 0 0 0 0
              1 0 0 0
            ⊕ 1 0 0 1
                0 0 1 0
              ⊕ 0 0 0 0
                  0 1 0 0
                ⊕ 0 0 0 0
                    1 0 0
```

3-bit CRC is 100.

e. See second paragraph of Section 7.3.1.

```
                     1 0 1 0 0
1 0 0 1 ) 1 0 1 1 0 1 0 0
        ⊕ 1 0 0 1
            0 1 0 0
          ⊕ 0 0 0 0
              1 0 0 1
            ⊕ 1 0 0 1
                0 0 0 0
              ⊕ 0 0 0 0
                  0 0 0 0
                ⊕ 0 0 0 0
                    0 0 0
```

Question 2

a. True or False
   i. TRUE.
   ii. FALSE – it does not incorporate RARP.
   iii. FALSE – they are 16-bits long.
   iv. TRUE.

b. See Section 6.5 and activities 6.1 and 6.2.

c. See Section 6.7.

d. See Section 5.3.

e. See Section 5.5.2.

Question 3

a. True or False
   i. TRUE.
   ii. TRUE.
   iii. FALSE – propagation delay is directly proportional to distance, transmission delay is inversely proportional to the data rate.
   iv. FALSE – it is an asymmetric connectionless protocol.

b. See Section 2.2.

c. See Section 3.3.

d. See first bullet points in Section 4.3.

e. See Section 4.5.

Notes

Appendix C

List of acronyms

ABM	Asynchronous Balanced Mode
AC	Alternating Current
ACK	Acknowledgement
ADSL	Asymmetric Digital Subscriber Line
AM	Amplitude Modulation
AMI	Alternate Mark Inversion
ANSI	American National Standards Institute
API	Application Programming Interface
ARM	Asynchronous Response Mode
ARP	Address Resolution Protocol
ARQ	Automatic Repeat Request
ASCII	American Standard Code for Information Interchange
ASK	Amplitude Shift Keying
ASN.1	Abstract Syntax Notation 1
AT	Attention
ATDM	Asynchronous Time Division Multiplexing
ATM	Asynchronous Transfer Mode
BCC	Block Check Character
bit	binary digit
bit/s	bits per second
BiSync	Binary Synchronous Communications
BnZS	Bipolar n Zero Substitution
BSC	Binary Synchronous Communications
CA	Certification Authority
CDMA	Code Division Multiple Access
CHAP	Challenge Handshake Authentication Protocol
CIDR	Classless Inter-Domain Routing
CMI	Code Mark Inversion
CMIP	Common Management Information Protocol
CRC	Cyclical Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
dB	deciBel
DAP	Directory Access Protocol
DC	Direct Current
DCE	Data Communications Equipment /Data Circuit-terminating Equipment

DES	Data Encryption Standard
DIB	Directory Information Base
DLE	Data Link Escape
DME	Differential Manchester Encoding
DMT	Discrete Multi-Tone
DNA	Digital Network Architecture
DNS	Domain Name System
DOS	Disk Operating System
DOCSIS	Data over Cable Service Interface Specification
DoD	Department of Defense
DSA	Directory Service Agent
DSLAM	Digital Subscriber Line Access Multiplexer
DSP	Directory Service Protocol
DTE	Data Terminal Equipment
DUA	Directory User Agent
DWDM	Dense Wave Division Multiplexing
EBCDIC	Extended Binary Coded Decimal Interchange Code
EIA	Electronic Industries Association
FDDI	Fibre Data Distributed Interface
FDM	Frequency Division Multiplexing
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FM	Frequency Modulation
FSK	Frequency Shift Keying
FTAM	File Transfer Access and Management
FTP	File Transfer Protocol
GUI	Graphical User Interface
HDB3	High Density Bipolar Order 3
HDLC	High-level Data Link Control
HFC	Hybrid Fibre-Coaxial
HTML	Hyper-Text Mark-up Language
HTTP	Hyper-Text Transfer Protocol
HTTPS	Secure HTTP
Hz	Hertz
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IMAP	Internet Mail Access Protocol
IP	Internet Protocol
IPCP	IP Control Protocol
IPsec	IP Security

IPX	Internetwork Packet Exchange
IrDA	Infra-red Data Association
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU-T	International Telecommunications Union – Telecommunications Standardization Sector
JPEG	Joint Photographic Expert Group
kbit/s	kilobit (thousand bits) per second
kHz	kiloHertz (thousand Hertz)
LAPB	Link Access Protocol Balanced
LAN	Local Area Network
LASER	Light Amplification by Stimulated Emission of Radiation
LCP	Link Control Protocol
LDAP	Light-weight Directory Access Protocol
LED	Light-Emitting Diode
LLC	Logical Link Control
LRC	Longitudinal Redundancy Check
MA	Multiple Access
MAC	Media Access Control
MAN	Metropolitan Area Network
Mbit/s	Megabits (million bits) per second
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MLT-3	Multi-Line Transmission 3-level
MPEG	Motion Picture Expert Group
ms	milliseconds (thousandth of a second)
m/s	metres per second
MHS	Message Handling Service
MHz	MegaHertz (million Hertz)
MOTIS	Message Oriented Text Interchange Standard
MPE	Manchester Phase Encoding
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
Mux	Multiplexer
mW	milliWatts (thousandth of a Watt)
NAK	Negative Acknowledgement
NCP	Network Control Protocol
NFS	Network File System
NIC	Network Interface Card
NPDU	Network Protocol Data Unit
NRM	Normal Response Mode

NRZ	Non-Return to Zero
NRZ-I	Non-Return to Zero - Invert
NRZ-L	Non-Return to Zero - Level
NSAP	Network Service Access Point
NSDU	Network Service Data Unit
NVT	Network Virtual Terminal
OSI	Open Systems Interconnection
OUI	Organisationally Unique Identifier
PAM	Pulse Amplitude Modulation
PAN	Personal Area Network
PAP	Password Authentication Protocol
PC	Personal Computer
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PKI	Public Key Infrastructure
PM	Phase Modulation
POP3	Post Office Protocol 3
PPP	Point-to-Point Protocol
PSK	Phase Shift Keying
PSTN	Public Switched Telephone Network
QAM	Quadrature Amplitude Modulation
RARP	Reverse Address Resolution Protocol
REJ	Reject
RFC	Request For Comment
RPC	Remote Procedure Call
RQ	Repeat Request
RTT	Round Trip Time
RZ	Return to Zero
SAP	Service Access Point
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit
SETI	Search for Extra-Terrestrial Intelligence
SHDSL	Symmetric High-bit-rate Digital Subscriber Line
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNMP	Simple Network Management Protocol
SNR	Signal to Noise Ratio
SPrings	Shared Protection Rings
SPX	Sequenced Packet Exchange
SSH	Secure Shell
SSL	cure Sockets Layer

STDM	Statistical Time Division Multiplexing
STP	Shielded Twisted Pair
TCM	Trellis Code Modulation
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TFTP	Trivial File Transfer Protocol
TIA	Telecommunications Industries Association
TLS	Transport Layer Security
TTL	Time To Live
TV	Television
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
VDSL	Very-high-speed Digital Subscriber Line
VRC	Vertical Redundancy Check
VT	Virtual Terminal
W	Watt (a unit that measures power)
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WWW	World wide web
XDR	External Data Representation
XML	Extensible Mark-up Language
XOR	Exclusive OR

Notes

Correction

Data communications and enterprise networking – volume 1

BSc in Computing and Information Systems

2005 2910222

August 2007, first correction

Chapter 2 – Basic concepts

p.15: Figure 2.5 Tree topology

The diagram of a tree toplology shown in the subject guide is actually another example of a bus topoplogy. Replace the diagram with the following one, which is a tree topology:

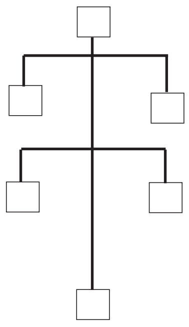


Figure 2.5: Tree topology

Chapter 5 – The network layer

p.52: 5.3.6 Error control – Internet checksum example

The 0 in the penultimate line should be in the rightmost column.

```
 Sent                                 Received
   1001  1011  0100  1001               1001  1011  0100  1001
   1110  1011  1101  0111               1110  1011  1101  0111
 ------------------------               0111  1000  1101  1110
 1 1000  0111  0010  0000             ------------------------
                        1             1 1111  1111  1111  1110
 ------------------------                                    1
   1000  0111  0010  0001             ------------------------
                                        1111  1111  1111  1111     Correct


 Checksum=
   0111  1000  1101  1110
                                        1001  1011  0101  1001
                                        1110  1011  1101  0111
                                        0111  1000  1101  1110
                                      ------------------------
                                      0 0000  0000  0000  1110
                                                             0
                                      ------------------------
                                        0000  0000  0000  1110     Error
```

p.55: 5.5.1 Encapsulation

In the second paragraph starting 'The option field', the second sentence should read 'The MSS excludes the transport, network and data link layer headers.'

p.58: 5.7 Transport Layer Security

The RFC reference in the margin note should be to RFC 4346.

p.59: 5.7 Transport Layer Security

In the fifth paragraph starting 'Secure authenticated communications', the second and third sentances should read 'The message is first decrypted using the client's private key and then encrypted using the server's public key.

At the server the message is decrypted using the server's private key and then encrypted using the client's public key.'

Chapter 7 – The data link layer

p.81: 7.3.1 Encapsulation

In the second paragraph starting 'Bit stuffing is somewhat similar', the bit string at the end should have 5 ones (11111) and not 6 ones.

p.82: 7.3.6 Error control

In the first paragraph, the first sentence should read 'Most data link protocols provide some **form** of error control'.

p.88: 7.3.6.2 Error Correction – Hamming Code – Example 1

In the third paragraph starting 'We can see that the error occurred in parity bits 1, 2 and 4 but not in 8', the byte at the very end of this paragraph should read 01101110.

Chapter 8 – The physical layer

p.101: 8.3.1.2 Types of physical media

In the last sentance of the first paragraph, the reference to Chapter 1 should have been to Chapter 2.

p.104: 8.3.2 Synchronisation

In the final sentance of the final paragraph, the reference to Section 8.3.5.4 should have been to Section 8.3.6.4.

p.107: 8.3.5 Error control – Nyqyist's theorem

In the fifth paragraph starting 'As an example', the problem should have stated that there were 32 and not 16 signalling levels and the note should have said that the bit rate was 31,000 bit/s and the Baud rate was 6,200 Baud.

This complete section is repeated below:

As an example, consider a 3.1 kHz telephone circuit in which a signal is carried using 32 signalling levels. Nyquist's theorem gives the maximum theoretical capacity of the channel as:

$$\begin{aligned} C &= 2B \log_2 V \\ &= 2 \times 3{,}100 \log_2 32 \\ &= 6{,}200 \times 5 \\ &= 31{,}000 \text{ bit/s} \\ &= 31 \text{ kbit/s} \end{aligned}$$

Note: This example illustrates the difference between the bit rate which is the quantity of data transmitter per second, in this case 31,000 bits per second, and the Baud rate, which is the number of signalling level changes or symbols per second. In this example the Baud rate would be 6,200 Baud.

Appendices

p.121: Appendix B: Model answer to Chapter 7 Question a. i.

The answer should read: FALSE – they are 48 bits long. The first 24 bits indicate the card manufacturer and the last 24 bits represent a serial number.

p.128: Appendix C: List of acronyms

SSL stands for Secure Sockets Layer.

Notes

Comment form

We welcome any comments you may have on the materials which are sent to you as part of your study pack. Such feedback from students helps us in our effort to improve the materials produced for the International Programmes.
If you have any comments about this guide, either general or specific (including corrections, non-availability of Essential readings, etc.), please take the time to complete and return this form.

Title of this subject guide: ....

....

Name ....

Address ....

....

Email ....

Student number ....

For which qualification are you studying? ....

Comments

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

Please continue on additional sheets if necessary.

Date: ....

Please send your completed form (or a photocopy of it) to:
Publishing Manager, Publications Office, University of London International Programmes, Stewart House, 32 Russell Square, London WC1B 5DN, UK.