

Protocols in multi-service networks



Protocols in multi-service networks



OpenLearn

Free learning from
The Open University

About this free course

This free course provides a sample of postgraduate study in Computing & IT

<http://www.open.ac.uk/courses/find/computing-and-it>

This version of the content may include video, images and interactive content that may not be optimised for your device.

You can experience this free course as it was originally designed on OpenLearn, the home of free learning from The Open University:

<http://www.open.edu/openlearn/science-maths-technology/computing-and-ict/systems-computer/protocols-multi-service-networks/content-section-0>.

There you'll also be able to track your progress via your activity record, which you can use to demonstrate your learning.

The Open University, Walton Hall, Milton Keynes, MK7 6AA.

Copyright © 2016 The Open University

Intellectual property

Unless otherwise stated, this resource is released under the terms of the Creative Commons Licence v4.0 http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_GB. Within that The Open University interprets this licence in the following way:

www.open.edu/openlearn/about-openlearn/frequently-asked-questions-on-openlearn. Copyright and rights falling outside the terms of the Creative Commons Licence are retained or controlled by The Open University. Please read the full text before using any of the content.

We believe the primary barrier to accessing high-quality educational experiences is cost, which is why we aim to publish as much free content as possible under an open licence. If it proves difficult to release content under our preferred Creative Commons licence (e.g. because we can't afford or gain the clearances or find suitable alternatives), we will still release the materials for free under a personal end-user licence.

This is because the learning experience will always be the same high quality offering and that should always be seen as positive – even if at times the licensing is different to Creative Commons.

When using the content you must attribute us (The Open University) (the OU) and any identified author in accordance with the terms of the Creative Commons Licence.

The Acknowledgements section is used to list, amongst other things, third party (Proprietary), licensed content which is not subject to Creative Commons licensing. Proprietary content must be used (retained) intact and in context to the content at all times.

The Acknowledgements section is also used to bring to your attention any other Special Restrictions which may apply to the content. For example there may be times when the Creative Commons Non-Commercial Sharealike licence does not apply to any of the content even if owned by us (The Open University). In these instances, unless stated otherwise, the content may be used for personal and non-commercial use.

We have also identified as Proprietary other material included in the content which is not subject to Creative Commons Licence. These are OU logos, trading names and may extend to certain photographic and video images and sound recordings and any other material as may be brought to your attention.

Unauthorised use of any of the content may constitute a breach of the terms and conditions and/or intellectual property laws.

We reserve the right to alter, amend or bring to an end any terms and conditions provided here without notice.

All rights falling outside the terms of the Creative Commons licence are retained or controlled by The Open University.

Head of Intellectual Property, The Open University
The Open University
United Kingdom by The Charlesworth Group, Huddersfield

Contents

Introduction	5
Learning Outcomes	6
1 Overview	7
1.1 Abbreviations used within this unit	7
1.2 Protocols in multi-service networks: introduction	8
2 OSI reference model	14
2.1 Layers of communication	14
2.2 Vertical communication	17
2.3 Horizontal communication	19
2.4 Examples of layer functions	20
3 TCP/IP protocol architecture	23
3.1 What does TCP/IP protocol architecture do?	23
3.2 Domain name system	25
3.3 Hypertext transfer protocol (HTTP)	26
3.4 Transmission control protocol (TCP)	29
3.5 Internet protocol (IP)	36
4 ATM protocol architecture	44
4.1 What is ATM protocol architecture?	44
4.2 ATM layers	46
4.3 IP over ATM	48
Conclusion	53
Keep on learning	56
References	56
Acknowledgements	58

Introduction

People have always communicated with each other – initially by face-to-face communication through gestures and sounds, then over a distance through written messages and signals in the form of fires, lights or flags. Technology, for instance in the form of electrical signals, has reduced many of the limitations of distance. Communication networks have become very important, and modern society depends on them for the smooth operation of economic and social activities. In this unit we regard a communication network as the means of interconnecting devices so that two-way communication is possible, and we shall focus on networks that interconnect telephones or computers. However, you should bear in mind other forms of network such as television and radio networks, which are primarily one-way, broadcast networks. In the future the separation between types of network may be less clear.

This OpenLearn course provides a sample of postgraduate study in [Computing & IT](#)

Learning Outcomes

After studying this course, you should be able to:

- evaluate technical descriptions of communication protocols and demonstrate an understanding of their operation
- describe the characteristics of circuit-switched and packet-switched networks, and of connectionless and connection-oriented modes in packet-switched networks
- describe the role played by primitives in the OSI reference model
- explain how 'vertical' and 'horizontal' communication takes place in the OSI reference model
- describe the main functions of the principal protocols in the TCP/IP architecture.

1 Overview

1.1 Abbreviations used within this unit

While reading this subject you may come across the following abbreviations.

AAL – ATM adaptation layer

API – application program interface

ARP – address resolution protocol

ATM – asynchronous transfer mode

B-ISDN – broadband ISDN

CLP – cell loss priority

DNS – domain name system

FTP – file transfer protocol

HTTP – hypertext transfer protocol

IANA – Internet Assigned Numbers Authority

ICT – information and communication technology

IEEE – Institute of Electrical and Electronics Engineers

IETF – Internet Engineering Task Force

IP – internet protocol

IPv4 – internet protocol version 4

ISDN – integrated services digital network

ISO – International Organization for Standardization

ISP – Internet service provider

ITU-T – International Telecommunication Union – Telecommunication Standardization Sector

LAN – local area network

OSI – open systems interconnection

POTS – plain old telephone service

PPP – point-to-point protocol

RFC – request for comments

RTP – real-time transport protocol

SAP – service access point

SDH – synchronous digital hierarchy

TCP – transmission control protocol

UDP – user datagram protocol

URL – uniform resource locator

WWW – world wide web

1.2 Protocols in multi-service networks: introduction

Early automatic telephone networks were built to carry only voice traffic and to provide a very simple telephone service – now called plain old telephone service (POTS). When computer networks started to appear, either they were separate from telephone networks or the data carried between computers was a small proportion of the traffic on the telephone network. There are various estimates for the growth of voice and data traffic, and various dates have been given for when data traffic will exceed voice traffic.

Figure 1 illustrates the relative growth rates of voice and data traffic from 1996 to 2005. It assumes that data traffic exceeds voice traffic during 2002 (Coffman and Odlyzko, 1998), and that voice traffic grows at a rate of 5 per cent per year and data traffic at a rate of 80 per cent per year. Because it is difficult to get accurate values for traffic, in the figure traffic is shown relative to the level of voice traffic in 1996 rather than as absolute levels. The rapid increase in data traffic can cause problems on telephone networks that were specifically designed to carry voice networks.

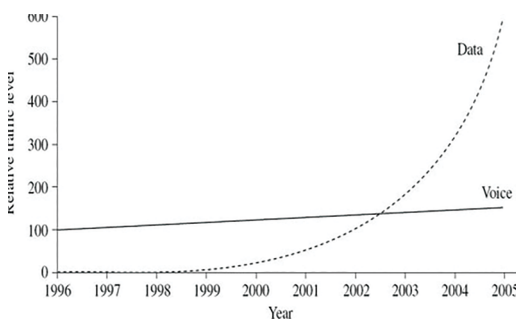


Figure 1 Voice and data growth

In many cases it makes economic sense for a single physical network to carry both voice and data traffic. In the 1980s this led to the publication of the requirements for an *integrated services digital network* (ISDN). Although ISDNs do exist in many countries, they have never become widespread. At the time of writing (2002), the requirements of broadband ISDNs (B-ISDNs) are being published, which extend the services of ISDNs to include high data rates. All you need to appreciate about ISDNs at this stage is the concept of the traffic being divided into specific categories, called services, according to the communication requirements of the data: for example, the type of data, the transfer mode and the transfer rate. An ISDN is an example of a *multi-service network*, which can be loosely defined as a network that provides a range of services over a common transport mechanism – that is, a common means of transferring data between devices. This may mean that different services receive different treatment to ensure that certain assurances about quality are fulfilled.

The Internet may be regarded as another example of a multi-service network, although the quality of service may not meet users' requirements for some applications. Private networks that operate to the same specification as the Internet can offer users a better quality of service, and the network operators can exercise greater control over the traffic. Such networks are called intranets.

In traditional telephone networks, a reserved transmission path is established between terminals before the users can talk to each other. Typically, the transmission path is capable of transmitting at a data rate of 64 kbit/s simultaneously in both directions. Because only one person usually talks at any one time and there are natural gaps in conversation, over half the transmission capacity is wasted. However, because the

transmission path is reserved, once a connection is established there is no queuing delay waiting for resources to become available. In addition, the end-to-end delay is constant because, once a path is set up, it does not change throughout a call. For voice traffic it is important that any delay in transport is constant as well as being as small as possible.

The transfer mode in which a path is reserved exclusively for a single communication is called *circuit switching*, and the type of service in which a connection has to be established before the exchange of data (whether for voice or computer applications) is called *connection oriented*. As you will see later in this section, these terms are not quite synonymous: circuit switching implies a connection-oriented service, but a connection-oriented service does not necessarily imply a circuit-switched transfer mode.

The requirement for a constant delay was not important in the early computer applications, and this encouraged the development of packet switching. In *packet switching* the message to be sent is divided into convenient groups of data, called packets, which are transferred independently over the network. Transmission capacity is not reserved. Instead, at each stage through the network, if transmission capacity is not available to send a packet immediately, the packet is stored until sufficient capacity becomes available, at which point it is forwarded on to the next stage. This process is sometimes called *store-and-forward*. Since capacity is not reserved for specific paths between users, if there are idle periods in the transmission of packets between two users, that capacity is available to other users and is not wasted.

However, waiting for transmission capacity to become available introduces a queuing delay in the transport of data. Moreover, because the amount of delay depends on the level of traffic from other users, which depends on factors that change with time, the delay will vary randomly and may differ for each packet.

In most cases, packet switching is more efficient than circuit switching at using transmission capacity, and it can be more resilient to network failures and congestion. For instance, if a switch detects that a link has become faulty, packets can be diverted to another link without any intervention by the users and without their knowledge. In circuit switching, if a link develops a fault, the transmission of data must be stopped and the circuit re-established. Although re-establishment may be performed automatically and very quickly, it may result in loss of data.

The type of packet switching described above is called *connectionless service*, or *datagram service* because packets are called datagrams in the internet protocol. A problem with this service is that there is no direct control over the level of traffic accepted by a network and, because the paths taken by packets can vary, the order in which they arrive can also vary. Where it is desirable to have more control over the transfer of packets, a connection-oriented service is used. The advantages of establishing a connection are that all the packets between the users follow a single path, and that if a switch has too much traffic it can refuse a request to set up a new connection. Note that the setting up of a connection does not necessarily reserve transmission capacity for a switch to forward a packet whenever it receives one, although actions may be taken which reduce the processing delay.

Figure 2 illustrates the differences between circuit switching and the two types of packet switching in the form of signal sequence diagrams. For each of the three cases the signals between four switches labelled S1 to S4 are shown. The figure gives just one example of transferring data for each mode. In practice, the efficiency of each mode will depend on the characteristics of the data being transferred, for example the amount and whether it arrives continuously or in bursts.

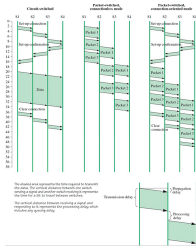


Figure 2 Signal sequence diagram for circuit switching and packet switching

Figure 2 also shows the transmission delay, propagation delay and processing delay for each signal transmitted between the switches:

- The *transmission delay* is the time between the first and last bits of a signal leaving a switch:

$$\text{transmission delay} = \frac{\text{number of bits}}{\text{data rate}}$$

- The *propagation delay* is the time taken for each bit to travel along the transmission medium:

$$\text{propagation delay} = \frac{\text{distance travelled}}{\text{velocity of propagation}}$$

- The *processing delay* is the time required for a switch to deal with the signal. It may include queuing delay in the case of packet-switched networks.

There are no commonly agreed definitions of the three types of delay introduced above, and you may see the terms in other sources with different meanings. For instance, the term ‘transmission delay’ is very often used to mean the total delay incurred in sending and receiving a message. The context in which the terms are used should indicate their meaning.

In [Figure 2](#) I have assumed that a switch does not start processing the next packet until it has finished transmitting the first. This is to simplify the diagrams. In practice, the receipt and transmission of packets may not require any intervention by a switch's processor. Also, the packet-switched, connectionless-mode example shows all three packets following the same path, which means that all packets arrive in the order in which they were transmitted; however, this is not necessarily always the case.

There is another type of switching in which a connection is not established explicitly by the user but is established by switches in the network for reasons of efficiency. This is called *flow switching*. Packet switches automatically detect a flow of packets between two devices. Once a flow has been detected, a connection is set up between two compatible switches along the path taken by the packets, and a label identifying this connection is attached to each subsequent packet. This allows switches to forward packets by looking only at these labels, which simplifies the forwarding and reduces the time it takes. Another variation of the same basic principle is possible, whereby a group of switches agree between themselves a means of tagging packets to reduce the forwarding delay. Section 4 of this unit introduces a form of flow switching called label switching. Note that the term ‘forwarding’ is used in packet switching for the transfer of packets between switches.

Message switching is another form of packet switching, in which complete messages are transferred as single packets. An early example of message switching was a telegraph network that included message centres where an incoming message was punched on

paper tape; the paper tape was later transferred (switched) to a paper tape reader for retransmission to the final destination or the next message centre.

SAQ 1

Complete Table 1, which reviews features of connection-oriented and connectionless modes of packet switching.

Table 1: Features to be completed for SAQ 1

Feature	Packet switched connection-oriented	Packet-switched connectionless
Reserved transmission capacity		No
Data rate constantly available	No	
Store-and-forward		
Constant route for packets		
Connection set-up		
Variable delay		
Control of quality of service		

Answer

Table 2 contains my review of the features.

Table 2: Answer to SAQ 1

Feature	Packet-switched connection-oriented mode	Packet-switched connectionless mode
Reserved transmission capacity	Maybe ¹	No
Data rate constantly available	No	No
Store-and-forward	Yes	Yes
Constant route for packets	Yes	No
Connection set-up	Yes	No
Variable delay	Yes ²	Yes
Control of quality of service	Yes	Maybe ³

¹ Reservation of buffer space may take place at the time of connection set-up.

² The variability in delay may be lower for the connection-oriented mode because of greater control over the quality of service.

³ Some measures of quality of service may be provided on a packet-by-packet basis by giving priority to some types of packet over others. This means that some applications may be given a better service than others, but it is difficult to provide absolute guarantees of quality of service.

SAQ 2

Draw to scale a signal sequence diagram of the form of [Figure 2](#) that shows a packet sent between systems A and B via system C. You should assume the following:

- the data rate between systems A and C is 500 kbit/s
- the data rate between systems C and B is 1 Mbit/s
- the packet contains 200 bytes;
- the distance between A and C and between C and B is 100 km;
- the velocity of propagation is $2 \times 10^8 \text{ m s}^{-1}$;
- the processing delay at C is 0.2 ms;
- the processing at C does not start until the complete packet has been received.

Answer

See Figure 3.

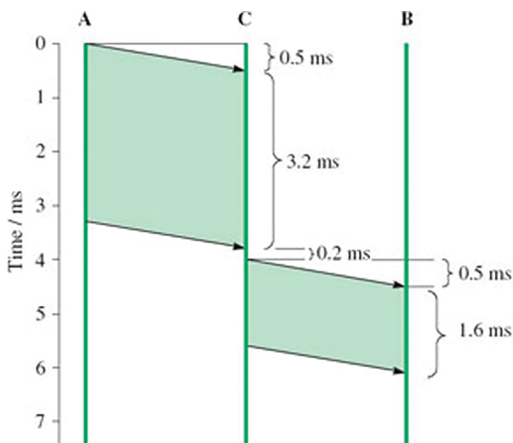


Figure 3 Answer to SAQ 2

The time required to transmit a packet of 200 bytes at a data rate of 500 kbit/s is given by:

$$\frac{200 \times 8}{500 \times 10^3} = 3.2 \times 10^{-3} \text{ s}$$

The time required to transmit a packet of 200 bytes at a data rate of 1 Mbit/s is given by:

$$\frac{200 \times 8}{1 \times 10^6} = 1.6 \times 10^{-3} \text{ s}$$

The time required for a signal to travel between A and C and between C and B is given by:

$$\frac{100 \times 10^3}{2 \times 10^8} = 0.5 \times 10^{-3} \text{ s}$$

The rest of this unit will look at a selection of communication protocols – the procedures that are essential for devices and systems to be able to communicate.

2 OSI reference model

2.1 Layers of communication

An *internetwork* is a network of networks, composed of terminals, switches and communication media. The overall objective of an internetwork is to allow communication between two (or more) networks. This simple description hides the complications that arise in real networks, in which the types of medium vary, transmission errors occur, transmission links fail, switches fail or become congested, equipment is produced by different manufacturers, networks are owned and maintained by different organisations, and so on.

In large internetworks, communication between systems is a complicated process, and to cope with this complexity the hardware and software in the systems are organised as a hierarchy of layers. Each layer performs some of the functions necessary to achieve communication between systems. The layers, particularly higher layers, are mostly implemented as software components of communication networks. It is very important to appreciate the hierarchical nature of communication systems: each layer, except the lowest, is built upon the layer below. In Figure 4 the functions are divided into four layers. This is just an illustration and different protocol architectures have different numbers of layers. Because one layer is built upon another, each layer provides a particular view of the communication system: that is, the layers provide *levels of abstraction* of the communication system. This may seem rather vague at present, but it should become more concrete later when we examine some practical examples.

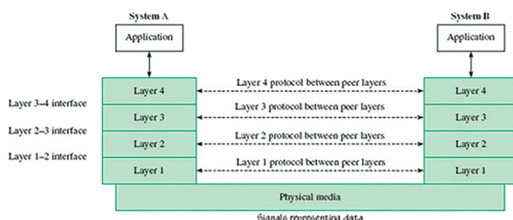


Figure 4 Layers in communication systems

The layered separation of functions can also be seen in everyday examples, such as sending a letter. You write a letter to a friend and enclose it in an envelope with the address written on it. You then post the letter and wait for it to be delivered. You don't give precise instructions about how the letter is to be delivered and what to do if unusual events occur. You leave these sorts of tasks to the postal service – the lower layers. The postal workers should not be interested in the contents of your letter – only the address written on the envelope – and your letter should be delivered with the contents unchanged, that is, free from error. The postal workers will decide the best route for your letter. This may involve carrying the letter by road, rail, sea or air but all this is irrelevant to you (although it may affect the cost of postage and can be considered as part of a quality-of-service agreement). Within the postal service the functions may be divided further. The local postal workers may be interested only in parts of the address to decide to which country or town your letter should be forwarded. Within a town, the postal code or street name may be important to allocate your letter to the appropriate delivery round. Perhaps

your friend has moved addresses and has told the postal service to forward their mail to a new address; the postal service would take care of this too.

I hope you can see that you and the postal workers are operating at different levels of abstraction. You may find it helpful to create your own analogies of layering. For instance, you could imagine the layers involved in someone dictating a letter to be translated into another language before or after delivery.

I need to explain what I mean by one layer being built on top of another. A layer can be thought of as providing *services* to the layer above. How these services are achieved is not the concern of the higher layer: it needs to know how to interact with the services, but not how they are implemented. The specification of the interface between two layers is very important because the equipment implementing the layers may be produced by different manufacturers. In providing services, a layer may perform several important functions. For instance, it is very convenient if the transfer of data can be considered to be free from transmission error. However, transmission errors do occur, and in some media they occur relatively frequently. It greatly simplifies the design of software if one layer performs the functions necessary to detect and correct transmission errors, thereby allowing all higher layers to assume that the transmission is free from this type of error.

A service normally requires some communication between systems in a communication network, and the set of rules which govern the communication is called a *protocol*. These rules are expressed in terms of the format of messages exchanged between two systems (the syntax), and the way in which the messages should be interpreted (the semantics). In this context, a protocol determines how communication takes place between the same layers in different systems, that is, between peer *layers*, and for this reason you may see this type of communication referred to as peer-to-peer. The peer layers exchange data as though there is a direct link between the two (as shown in [Figure 4](#)), but in reality all the data is passed down through all the layers and is carried by the physical media in the form of signals. However, it is convenient to imagine that there is a direct *virtual connection* and ignore all lower layers. It is very important to have open and internationally agreed protocols which enable widely different types of equipment to interact with each other.

In 1978 the International Organization for Standardization (ISO) defined a framework for describing the layers in communication networks, called the *Open Systems Inter-connection (OSI) reference model*. Although many networks may not fit the strict definition of the model and the model has been modified with the introduction of sub-layers, the OSI reference model does provide a very good framework for discussing communication networks.

The OSI reference model has seven layers; these are shown at each end system in Figure 5. The layers were chosen after considerable reflection and the choice was influenced by the following broad principles (Tanenbaum, 1996; ITU-T X.200, 1994):

1. There should not be more layers than is necessary.
2. Boundaries should be located where they have proved successful in the past.
3. Boundaries should be located to minimise the interactions between layers.
4. Boundaries should be located where a standardised interface may be useful.
5. Separate layers should be created to perform functions that are associated with different technologies or levels of abstraction.
6. Functions associated with similar technology should be collected together in the same layers.

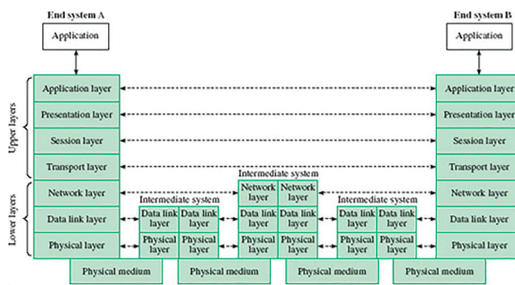


Figure 5 OSI layers

Before going into more detail about Figure 5, I shall describe very briefly the main functions of each of the seven layers:

1. Physical layer – provides the mechanical, electrical and procedural means for transmitting bits over a communication medium.
2. Data link layer – provides services for the transmission of data between directly connected systems in a communication network.
3. Network layer – handles the routing of data through communication networks.
4. Transport layer – provides reliable end-to-end services without being concerned about the route through communication networks.
5. Session layer – provides facilities to organise and synchronise dialogues, i.e. communications that consist of several strands such as audio and video components.
6. Presentation layer – deals with issues about how data is represented and ensures that the systems agree on how the information is transferred.
7. Application layer – provides the means for application programs to access the communication system represented by the OSI reference model. For instance, the application layer can provide services for supporting file transfer and email.

The lowest three layers are primarily concerned with the problems of transferring data across physical networks, and the highest four layers are associated with end-to-end issues and not the specific details of any communication network. Intermediate systems in Figure 5 are shown as pairs of stacks of layers. Different conditions may be encountered on the two sides of an intermediate system: for instance, different transmission media may link two systems together.

SAQ 3

Briefly summarise the view, or level of abstraction, each layer provides to its higher layer.

Answer

I have listed my version of the views below, but don't worry if you have included other points or you don't understand at this stage all the points I have included.

1. The physical layer deals with the practicalities of transmitting data over a physical medium and hides these from the data link layer. So from the data link layer's viewpoint transmitting data takes the form of sending 1s and 0s.
2. The data link layer may have to deal with errors occurring during transmission and with complications that arise from sharing a transmission medium. These problems are hidden from the network layer.

3. From the network layer's viewpoint a network sends blocks of data between switches, but these blocks may be lost or reordered. The network layer navigates the blocks of data through the network(s) and hides these complications from the transport layer.
4. The transport layer does not need to know about the physical structure, or architecture, of the network(s). It views a network as a direct channel between two users for transporting data.
5. The session layer does not always take any actions associated with the transport of data, but it may have to enhance the services of the transport layer in order to fulfil user requirements. For instance, a session may involve a sequence of independent transactions.
6. The presentation layer provides communication services to users about how certain information is represented, such as dates and people's names.
7. The application layer deals with specific classes of applications and may involve the identification of users.

Requests for service between adjacent layers take place through interfaces, which define the services provided by the lower layer and the operations necessary to interact with the services.

2.2 Vertical communication

Figure 6 shows the OSI view of adjacent layers. The interface between two layers in the same system is called a *service access point* (SAP). One of the features of a service access point is that it has an identifier, or an address, which allows each communication between adjacent layers to be uniquely identified. The processes that communicate across the interface are called *entities*. These are typically software routines, but may also be hardware components. The notation in Figure 6 may appear odd, but it is designed to apply to any pair of layers in the reference model. The upper layer is called $(N + 1)$ and the lower layer (N) . Layer (N) is the service provider and layer $(N + 1)$ the service user.

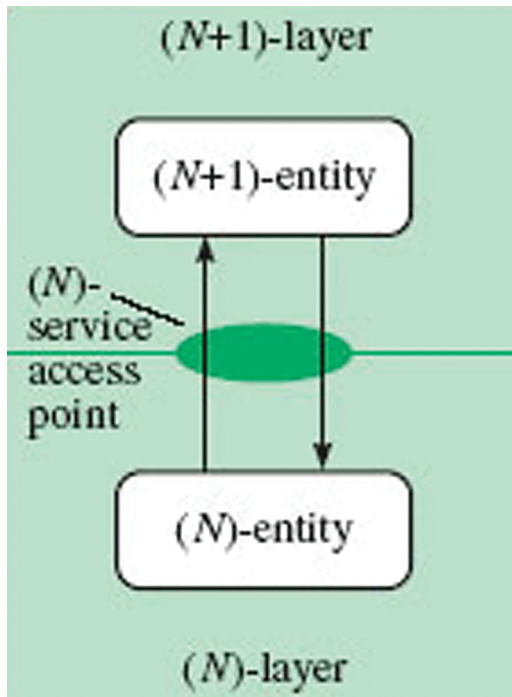


Figure 6 Adjacent layers in the OSI reference model

The interaction between adjacent layers is expressed in terms of issuing and receiving *primitives*. The form taken by primitives depends on the implementation of the system but typically they are software procedure calls. In two systems involved in any communication there is a range of primitives depending on the service required. For each primitive up to four basic types are available:

1. request – an entity invokes a service
2. indication – an entity is informed of an event
3. response – an entity reacts to an event
4. confirm – an entity is informed of the result of an earlier request.

Figure 7 illustrates an exchange of all four types of primitive, and some concrete examples are given below. Because primitives are issued and received across the boundaries between adjacent layers in the OSI reference model, their interaction is sometimes referred to as *vertical communication*.

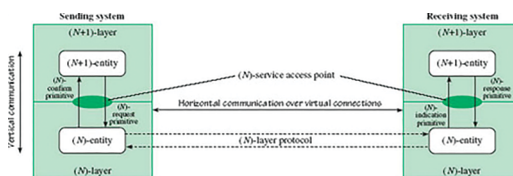


Figure 7 OSI types of primitives

Most primitives carry parameters, which provide the information necessary to fulfil the service. For instance, the service of setting up a connection between two OSI transport entities requires a connection primitive, and the request type of this primitive has the following parameters:

- called address;
- calling address;

- receipt confirmation selection;
- expedited data selection;
- quality-of-service parameter set;
- user data.

A procedure call implementation of a connection request may take the form of:

Call CONNECT-request (called address, calling address, receipt confirmation selection, expedited data selection, quality-of-service parameter set, user data)

The indication, response and confirm types of this connection primitive have similar sets of parameters. Once a connection has been established data is transferred by invoking a service with a data transfer primitive. This primitive has only two forms, request and indication, which have the parameters:

- user data;
- confirmation request.

The called and calling addresses were supplied when the connection was established and it is assumed that the entities still know this information.

SAQ 4

The equivalent data transfer primitive associated for a connectionless service needs to include parameters for the called and calling addresses. Why?

Answer

In a connectionless service, the flow of data is not associated with a connection, so no connection primitive is issued. Each block of user data must therefore be accompanied by the information necessary for its delivery.

At this stage it is not necessary to understand what information each of these parameters represents, merely to appreciate that primitives do carry additional information necessary for a layer to provide services.

2.3 Horizontal communication

In the OSI reference model there is a clear separation of services and protocols, but this separation is not always evident in practical applications, so it is worthwhile spending some more time on the differences between them. A service is provided by one layer to the layer above, and the capabilities of a service are defined in terms of primitives and their parameters. A service relates to two adjacent layers in the same system. In contrast, a protocol defines the communication between two peer entities in different systems. To complete a service request, an entity may need to communicate with a peer entity via a protocol, but the protocol chosen by a layer should not be apparent to the service user. You should bear in mind that the communicating systems may be quite different, and the protocols facilitate the transfer of information between them. Protocols define how data is transferred between systems and perhaps it is for this reason that the features of protocols tend to dominate discussions rather than the underlying services.

As [Figure 7](#) illustrated, the information an (N) -entity receives from an $(N + 1)$ -entity is transferred to an (N) -entity in another system by an (N) -layer protocol. Figure 8 shows in more detail how this is done. The data is placed in a data unit called a *protocol data unit*. This is simply a generic term for a packet that applies to all of the seven layers. A protocol data unit contains the service-related data from the $(N + 1)$ -entity, plus some *protocol control information* which is necessary to co-ordinate the transfer of data. This construction of protocol data units may be repeated in all the layers in the OSI reference model whenever data is transferred between two systems. The process of adding protocol control information to the data is called *encapsulation*. A very important point to remember is that the entity associated with the lower-layer protocol does not interpret the service-related data associated with the higher layer; to the lower layer, it is just data passed to or received from the higher layer. In Figure 8 the protocol control information is added as a single header to the left of the higher-layer data to illustrate the process of encapsulation, but in actual protocols the information may be placed anywhere in the protocol data unit. The communication between peer entities through a virtual connection in different systems is sometimes referred to as *horizontal communication*.

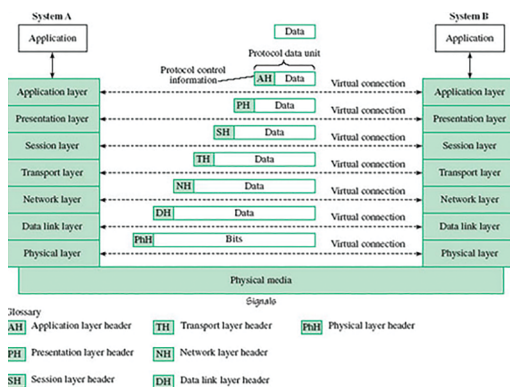


Figure 8 Encapsulation

2.4 Examples of layer functions

There are several functions that can be performed at one or more of the OSI layers. Some of the more common ones are discussed below.

Connection control

For connection-oriented services, a connection must be established between peer entities. A connection has three phases: connection set-up, data transfer and connection clear. If the peer protocol supports connections, each protocol data unit type corresponds to a primitive type; for instance, a connection request primitive is transported by a connection request protocol data unit. If the peer protocol does not support connections, the peer layers must perform additional functions to give the characteristics of a connection-oriented service.

Data flow

The user data is passed transparently between peer layers by calling on the service of the layer below. Network or end-system constraints may make it necessary to *control* the rate of the *flow* of protocol data units between peer layers.

Segmentation and re-assembly

If the data received from the higher layer is too large to fit in the maximum size of protocol data unit, the lower layer will have to split it into smaller chunks. This is called segmentation, and additional protocol control information is required to ensure the correct re-assembly of the data in the peer layer of the receiving system.

Sequencing

If the higher layer requires its data to be received in the order sent and the lower layer does not provide a service with this feature, the upper layer must preserve the order by performing some additional processing. This will typically involve including sequence numbers in the protocol control information.

Acknowledgement

A layer must take additional actions if its lower layer does not provide a sufficiently reliable means of transferring protocol data units. Typically, this involves including identifiers in the protocol control information and the receiving system explicitly acknowledging receipt of the protocol data units. Time-outs are frequently employed to prevent systems waiting indefinitely for acknowledgements to arrive. When a time-out occurs, the system either resends the protocol data unit or indicates that the connection may be faulty.

Error control

Transmission errors do occur in networks, causing bits to be received incorrectly. Protocol data units can also be lost, delayed or duplicated. A common technique for detecting bit errors is to include a cyclic redundancy check in the protocol control information. The basic principle of a *cyclic redundancy check* is that a check word is generated by the sending system and included in the protocol data unit. The receiving system generates a new check word based on the received data and compares this with the received check word; if they differ then errors have occurred. Any detected errors typically result in the protocol data unit being discarded.

Synchronisation

It may be necessary for two systems to agree on the occurrence of events. For example, two databases may have to agree that a record has been updated on both systems. Another example is when independent protocol data units may be delayed by different amounts during transfer, and the receiving system must retime the protocol data units to ensure that the same timing relationship exists in the received protocol data units as in the transmitted units.

Addressing and routing

In the context of the OSI reference model, an address identifies a service access point on the boundary of two layers. An address may be unique throughout an entire internetwork or may have only local significance – be unique within a single system, a single network or a section of a network. Some addresses, or parts of addresses, are assigned by a central authority to ensure that naming conventions are followed and the addresses are unique. Addresses at the network layer are very important because they identify systems in an internetwork. An address may refer to a group of systems as well as a single system and in the extreme case may refer to all the systems in a network.

Some intermediate systems (see [Figure 5](#)) will select the next link in the transfer path according to the addresses in protocol data units. The determination of the best link is part of the routing function of systems.

Multiplexing and demultiplexing

An entity may have to establish several connections in order to perform a service, or a connection in a lower layer may need to support several connections in a higher layer for reasons of efficiency. The function of sending protocol data units from several

connections over a single connection is called multiplexing. Demultiplexing is the reverse process, in which the data from protocol data units arriving on a lower-layer connection are distributed over several higher-layer connections.

SAQ 5

What are the response options of an entity in the sending system if an end system or an intermediate system discards a protocol data unit because it detects an error?

Answer

The entity in the sending system may have no way of knowing that the protocol data unit has been discarded, so it may do nothing. However, the entity may be expecting an acknowledgement and, if that does not arrive within a given period, it may decide to retransmit the protocol data unit or declare the connection to be faulty.

Even if the receiving entity does nothing, that may not be the end of the story because an entity in a higher layer may detect that some data is missing and cause one or more of its protocol data units to be retransmitted.

3 TCP/IP protocol architecture

3.1 What does TCP/IP protocol architecture do?

The *Internet* is a worldwide public internetwork, which allows computers to communicate with each other even though they may have different manufacturers and different operating systems. The origins of the Internet lie in a project of the US Defense Advanced Research Project Agency in the 1970s, where it was intended to foster communication between research institutions rather than operate for profit. However, a substantial amount of traffic carried by the Internet is now related to commercial activities in the form of online shopping, banking or entertainment, or to private emails.

The technical aspects of the Internet are controlled by an international group called the Internet Engineering Task Force (IETF). Information about the Internet is recorded in documents called RFCs (requests for comments). These documents are available online from several sites, such as the IETF. Some of the RFCs contain official definitions of protocols, but others contain proposals for new protocols or explanatory information about the Internet.

TCP/IP (transmission control protocol/internet protocol) is the name given to a suite of protocols designed for the Internet, but suitable for any internetwork.

The Internet contains the following broad classes of network, as shown in Figure 9:

- backbones – providing high-speed data links between networks (e.g. NSFNET, EBONE)
- regional networks – clusters of networks for specific organisations, such as universities (e.g. JANET)
- commercial networks – e.g. ISPs (Internet service providers) providing subscribers with access to the Internet or organisations providing their employees with access to the Internet;
- national networks – groups of networks covering specific countries;
- local networks – consisting of individual groups of computers such as those on a university campus.

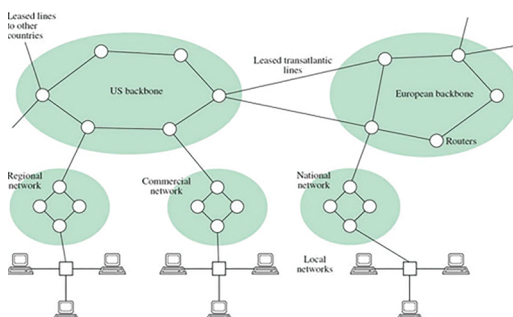


Figure 9 The Internet

Figure 10 shows the TCP/IP reference model and Figure 11 shows examples of protocols from various organisations and their association with the TCP/IP protocol suite. The

model in Figure 9 follows a layering principle similar to that advocated by the OSI reference model, but only four layers are defined.

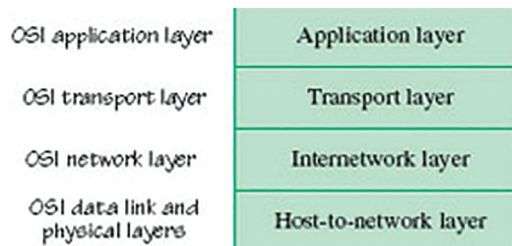


Figure 10 TCP/IP reference model

You may come across slightly different TCP/IP reference models from that shown in Figure 10. When the TCP/IP protocol suite was developed, although the concept of layering was important, the emphasis was on producing practical protocols. As a consequence, the TCP/IP reference model reflected these protocols, rather than the protocols being produced according to a specific reference model as was the case with OSI protocols.

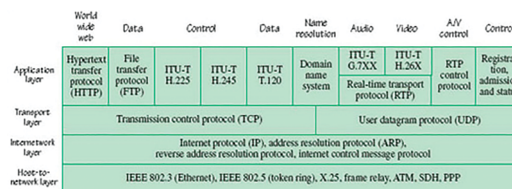


Figure 11 Example protocols

The application layer (see Figure 11) offers standard services to application software running on systems. Access to these services is typically provided in the form of *application program interfaces* (APIs), which in effect are procedure call implementations of the primitives that I described in the context of the OSI reference model.

The transport layer provides end-to-end communication over an internetwork. Both connectionless and connection-oriented modes of transfer are available through the UDP (user datagram protocol) and TCP (transmission control protocol) protocols respectively. The internetwork layer deals with the functions associated with specific networks. The main protocol in this layer is IP (internet protocol) and the other protocols in Figure 11 for this layer support the operation of IP. The internetwork layer shields the transport layer from the difficulties of locating systems in communication networks and transferring data across possibly disparate networks. At this layer, packet switching is performed by switches called *routers*. A router will use the destination address of the data to decide:

- whether the destination system is connected directly to it – in which case the data can be delivered directly;
- to which interface the data should be forwarded if the destination system is not connected directly to it.

The host-to-network layer deals with the reliable transfer of data between nodes in a communication network. Protocols in this layer are not strictly within the TCP/IP protocol suite because they are controlled by other standards bodies and have uses outside the Internet context. A host is an Internet term for any device or computer system connected to the Internet that has an Internet address, and in general refers to the terminating devices or end systems.

Figure 12 shows an example of transferring files between two hosts using the file transfer protocol (FTP) in an internetwork. The hosts transfer data over three distinct physical networks: two local area networks and one public data network. The specific types are not important here; you need only appreciate that different types are possible. The protocol data units for the FTP and TCP protocols are transferred transparently by the lower-level protocols and are not affected by the choice of the physical networks. The IP protocol exchanges messages between hosts A and B and the two routers. In this example different host-to-network layer protocols are employed in the different physical networks to carry the IP protocol data units.

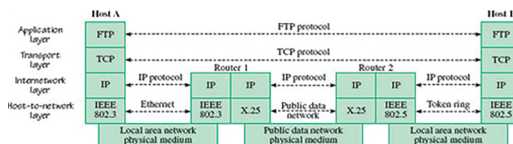


Figure 12 Example of TCP/IP communication

3.2 Domain name system

Applications use easy-to-remember names for hosts on the Internet, but before sending any data to a host an application in the source host must translate its name for the destination host to the numerical network address.

The Internet is divided into domains, and an authority in each domain is responsible for allocating names. However, the domains may be divided into sub-domains and the responsibility of allocating sub-domain names may be delegated to other authorities. In this way the names form a tree structure as shown in Figure 13. The Internet Assigned Number Authority (IANA) has overall responsibility for assigning domain names on the Internet.

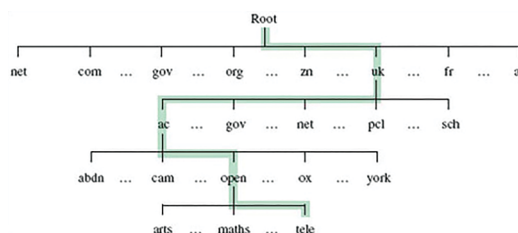


Figure 13: Tree structure of names on the Internet

The full names of hosts are written as a sequence of words, separated by full stops, and each word refers to a domain or sub-domain. The names are written from left to right, with the most specific (the host) on the left and the top-level domain on the right. For example 'tele.open.ac.uk' is the full name of a host called 'tele' in the sub-domain 'open', which is itself within the larger sub-domain called 'ac' (for academic). The top-level domain for this host is 'uk', which is a geographical domain name. Broadly speaking, there are two classes of domain name – organisational and geographical. You can see from Figure 13 that some organisational names are the same as national names, illustrating the point that the individual domain names are not necessarily unique, but the full host names are. The delegation of naming authority greatly simplifies the allocation of names because the naming authorities in higher domains do not need to be notified about changes to names

in the lower domains. However, naming authorities must be able to determine the physical location of hosts in their immediate sub-domain.

The network address determines the path through an internetwork, and the translation from a host name to a network address is typically done by the application invoking a library program called the *resolver*. In principle, the resolver program sends messages to a database which contains the network addresses corresponding to the domain names. It would not be practicable, however, to have a single database for the Internet, so distributed databases, called the *domain name system* (DNS), hold the translation information.

Each domain has one or more name servers and, whenever a host has a name to resolve, it sends the host name to a local name server. If the local name server has a record for the name it can reply immediately with the corresponding network address. If not, the local name server passes the query to a name server for a higher-level domain. In the worst case this would be a name server for the top-level domain as shown in Figure 14.

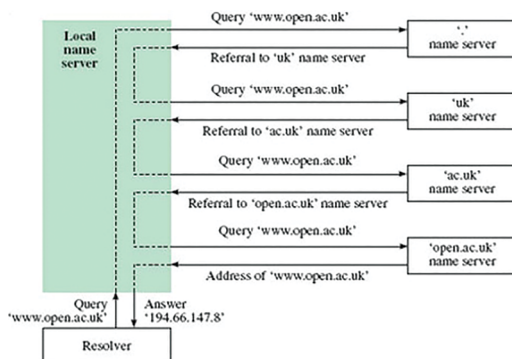


Figure 14 Resolving domain n

The top-level domain name server will at least know the name servers of all its immediate sub-domains, and can supply the local name server with the network address of a name server in the appropriate sub-domain. The local name server then repeats the query to this name server. The process is repeated until the query is satisfied.

Another method of resolving a host name is for the queried name servers to pass on the query directly to the next name server rather than passing the address of the next name server back to the local name server to continue the process.

3.3 Hypertext transfer protocol (HTTP)

In this section, I shall look at one example of an application of the TCP/IP protocol suite – sending hypertext pages over the *world wide web* (*WWW* or simply the *web*). However, first I shall very briefly summarise the main features of the web that are relevant to this discussion. There are many sources of information about the web on the web itself for those who want to know more.

In very basic terms, the web is an application of the Internet for accessing resources wherever they are located. Each resource on the web is found through a name called a *uniform resource locator* (URL). A URL consists first of the name of the scheme for communicating with the resource at the application layer (e.g. FTP, HTTP, MAILTO) followed by a scheme-specific part. The only scheme we shall look at here is HTTP, and the syntax for HTTP URLs takes the form of:

`http:// <host> : <port> / <path> ? <search-part>`

where the names in angled brackets indicate that values are inserted in actual URLs. The 'host' is the domain name as given in the domain name system of the Internet. The 'port' is optional and is described in the next section, but it can be thought of as a way of identifying the different applications that may be running in an application layer. The 'path' in a URL identifies the name of the resource in the host's filing system. The 'search-part' is optional and is used when the resource is a search engine. If no 'port' value is given, then a default value is used.

SAQ 6

Identify the component parts of the following URL:

`http://www.altavista.digital.com/cgi-bin/query?q=multi-service+network`

Answer

The name of the host is 'www.altavista.digital.com', the name of the path is 'cgi-bin/query' and the search-part is 'q=multi-service+network'. No port value is specified, so a default value is used.

When the resource on the web is a hypertext page with links to other pages, retrieving one page of information may trigger several subsequent requests to obtain resources.

In many situations, users access the web through a *proxy server*, which separates an internal network from the Internet and thereby can offer some degree of protection to the users. A proxy server can also provide temporary local storage (a cache store) of resources obtained from the web. Storing a resource locally can avoid repeating a recent request for the same resource.

Figure 15 shows the application layer view of the communication between peer entities. In this simple example the only function of the proxy server that we are interested in is that of repeating the messages it receives.

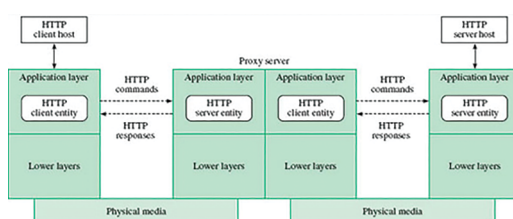


Figure 15 HTTP protocol model

HTTP (*hypertext transfer protocol*) takes the form of commands from a client, typically a browser computer program, and responses from a server. A command (called a method in HTTP terminology) can occupy several lines. The first line has the name of the command, the identity of the resource, and the scheme version number. Subsequent lines may contain parameters, which modify the command, and any data to be transferred to the server. Examples of parameters are the name of the host and a date giving the last time a resource was modified. A response from a server takes the form of a status line, additional information about the type of response, and the resource itself.

Commands and responses are defined in RFC 2616. Some example commands are listed in Table 3 and example responses in Table 4.

Table 3: Example HTTP commands

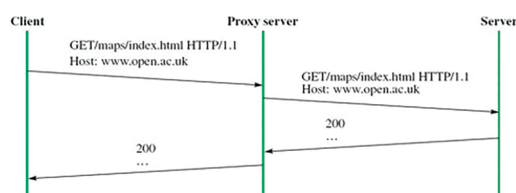
Command	Description
Connect	Establishes a tunnel ¹ through a proxy server
Delete	Requests the removal of a resource
Get	Requests the retrieval of a resource
Head	Requests the retrieval of information about a resource rather than the resource itself
Options	Requests information about the capabilities of a server or the requirements of a resource
Post	Requests that a resource accepts some information
Put	Requests that a resource is stored at the given location
Trace	Requests that a server takes part in some test

¹A tunnel is created by encapsulation. A received message is encapsulated in another message so that the received message is transferred transparently through a node or network.

Table 4: Example HTTP responses

Response	Description
100	Continue
200	OK
202	Accepted
302	Found
400	Bad request
404	Not found
406	Not acceptable

Figure 16 shows an example of commands and responses between an HTTP client and server via a proxy server.

**Figure 16 Example HTTP communication**

HTTP protocol data units are transferred by requesting a connection-oriented service from the transport layer, for example by calling on the TCP protocol to transfer messages between the client and server. In the first version of HTTP, individual connections were set up for each resource and this could entail establishing several consecutive connections for hypertext pages that included links to other resources. The current version of HTTP has improved efficiency by allowing a host to maintain a connection for more than one

transfer if appropriate. Efficiency is further improved by allowing a host to submit several requests without waiting for an acknowledgement.

3.4 Transmission control protocol (TCP)

As I outlined in the previous section, peer entities in clients and servers exchange HTTP protocol data units when they wish to transfer a resource over the web. I gave very little detail about this because I wanted to focus on the general features of protocols in the application layer of the TCP/IP model. The HTTP protocol data units are transferred from the sender host to the receiver host by calling on the services of the transport layer. In the case we are considering, the transport layer provides the services through the *transmission control protocol* (TCP).

A host may have several applications active simultaneously and it is necessary to separate the messages arriving from the transport layer. In the TCP/IP model this is achieved through port numbers. Port numbers are stored as 16-bit numbers, so have the range 0 to 65535 ($65535 = 2^{16} - 1$). Those in the range 0 to 1023 are reserved for standard services and are called *well-known port* numbers. Those in the range 1024 to 49151 refer to ports registered to provide services to unknown callers. The remainder, 49152 to 65535, refer to dynamic or private ports and may be used as required. Well-known port numbers are embedded in the software, so for standard services a host does not have to take any special actions to find out which port number to specify.

SAQ 7

Which of the layer functions mentioned in Section 2 is accomplished through port numbers in the TCP protocol?

Answer

Multiplexing and demultiplexing. Port numbers allow several applications to establish separate virtual connections to different applications.

The receiver host has the port number 80 reserved for receiving HTTP messages, but for certain other applications the sender host can select any port number it wishes in the range 1024 to 65535.

Before I describe the primitives issued to support HTTP, we need to look at TCP to understand some of its functions. The main objective of TCP is to provide a reliable end-to-end byte stream over an internetwork. The TCP entity receives a byte stream from an application entity and divides the data into units not exceeding 65535 bytes. In practice the size is about 1500 bytes because of the maximum transfer unit supported by the underlying physical networks. These protocol data units are then passed down to the internetwork layer for delivery to the destination host. The transport layer at the destination re-assembles the data units into a byte stream for delivery to the application layer in that system.

Some insight into the operation of TCP can be obtained by examining the format of the protocol data units, which are called *segments* in TCP terminology, as shown in Figure 17.

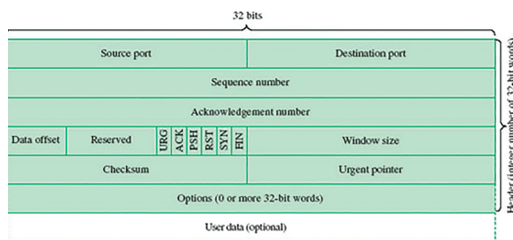


Figure 17 TCP segment format

Figure 17 identifies the following fields in a TCP segment:

- source port (16 bits) – identifies the port number of the source;
- destination port (16 bits) – identifies the port number of the destination;
- sequence number (32 bits) – gives the number of the first data byte in this segment, unless the synchronise flag is set, in which case the first data byte is this number plus 1;
- acknowledgement number (32 bits) – if the acknowledgement flag is set, this is the sequence number of the next data byte that the sender of this segment expects to receive;
- data offset (4 bits) – gives the number of 32-bit words in the header and therefore indicates where the data begins in the segment;
- reserved field (6 bits) – these bits have no defined purpose;
- URG (1 bit) – the urgent flag indicates that the urgent pointer field is significant;
- ACK (1 bit) – the acknowledgement flag indicates that the acknowledgement number field is significant;
- PSH (1 bit) – the push flag indicates that the receiver is to deliver all data to the application rather than wait for a buffer to fill;
- RST (1 bit) – the reset flag indicates that the connection is to be reset because some problem has arisen;
- SYN (1 bit) – the synchronise flag indicates that the initial sequence number of the data bytes is to be the value of the sequence number field plus 1; in effect it indicates a request to establish a new connection;
- FIN (1 bit) – the finish flag indicates that there is no more data from the sender and also indicates a request to clear the connection;
- window size (16 bits) – identifies the number of data bytes that the sender of this segment is willing to accept without acknowledgements;
- checksum (16 bits) – this is a simple check sequence of all the 16-bit words in the header and user data fields;
- urgent pointer (16 bits) – if the urgent flag is set, this is the offset in the data field of the first data byte that follows the urgent data, i.e. in effect it gives the position of the end of the urgent data;
- options (0 or more 32-bit words) – a list of option parameters, e.g. the maximum segment size;
- user data (optional) – user data to be transferred transparently between application entities.

Notice that the network address of the host is not included in any field of a TCP segment. This is because the network address is important to the internetwork layer rather than the transport layer and, as you will see later, appears in the IP protocol data unit.

SAQ 8

Why is data optional in TCP segments?

Answer

As you will see in the next section, some TCP segments do not transfer data from higher layers but perform functions such as setting up a virtual connection.

3.4.1 Connection control

TCP does not assume a reliable communication network and takes precautions against the loss or duplication of protocol data units. One method is the *three-way handshake* procedure for establishing a connection between two hosts. Figure 18 shows the TCP procedure for this. This connection is a virtual connection of the type described in Section 2; many virtual connections can share a physical communication channel.

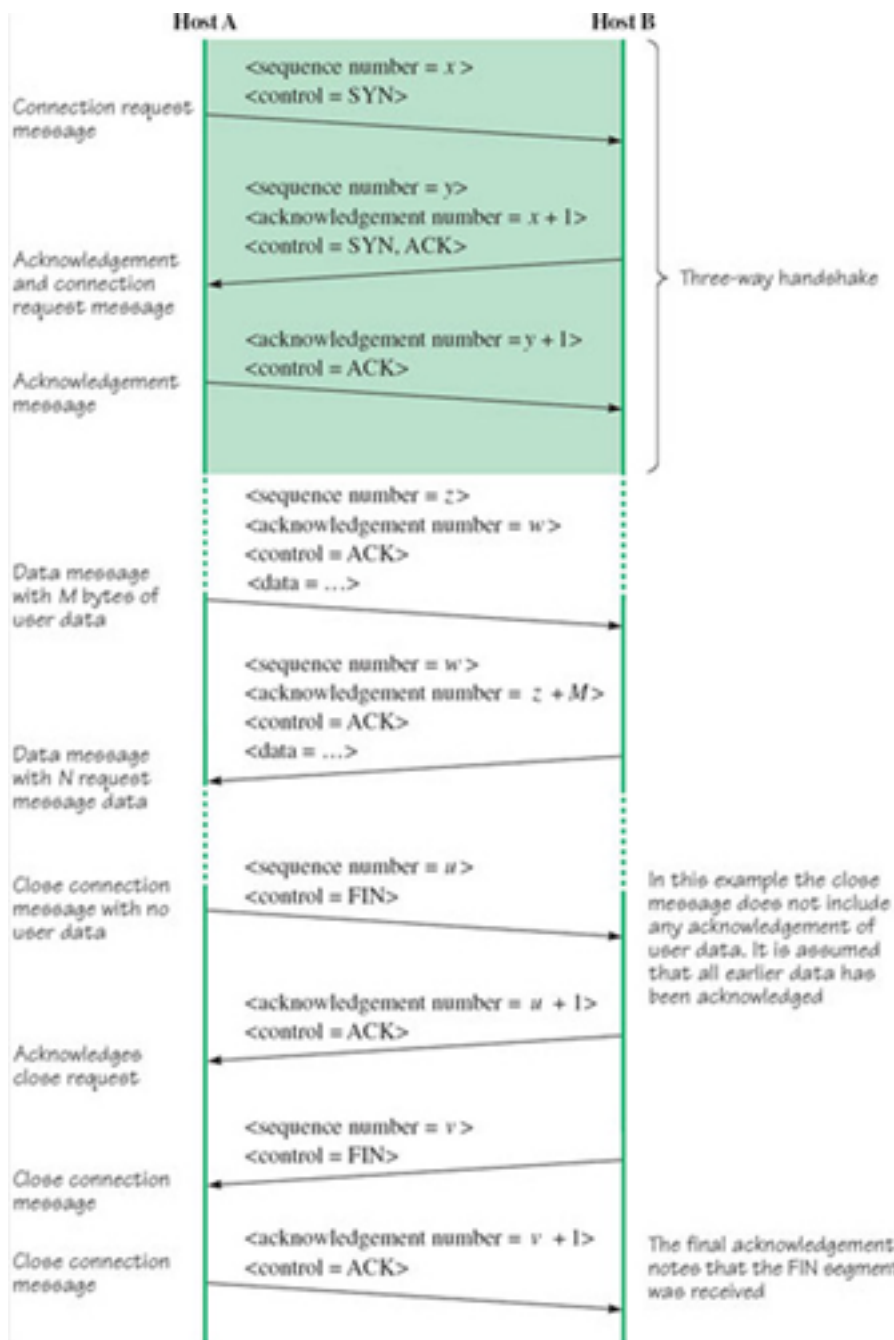


Figure 18 TCP three-way handshake procedure

The condition of the control flags in TCP segments determines how a host will interpret the segments it receives. The first protocol data unit (segment) from host A to host B is in effect a 'connection request message'. This is indicated by the SYN flag being active. The value in the sequence number field indicates the initial sequence number, x , which means that the sequence number of the first data byte that A is going to send to B is $x + 1$. The value of x is chosen by A to minimise the risk of duplicated sequence numbers. Host B responds by sending an 'acknowledgement and connection request message' to A. This is indicated by both the SYN flag and the ACK flag being active. The acknowledgement number field contains the value $x + 1$, which is the first sequence number host B expects to receive. The initial sequence number chosen by B to transmit its data in this example is represented by the letter y . In the final stage of the three-way handshake, host A sends an 'acknowledgement message'. This message has the ACK flag active and the acknowl-

edgement number equals $y + 1$. Figure 18 shows that this is followed by user data sent between the two hosts.

Although not shown in Figure 18, the segments that set up a connection may contain user data in the data field, but the receiving host should not deliver this data until the connection has been established.

Figure 18 also shows an example of closing a connection in both directions. Host A first requests the connection to be closed. Host B acknowledges this request and in a separate message requests that the connection in the opposite direction is closed also. Notice that the values of the acknowledgement and sequence number fields make certain that there is no ambiguity about how much data has been received by each host.

3.4.2 Primitives

TCP primitives are implemented as application program interfaces (APIs). Those provided on a particular computer system depend on how the TCP software is written. I shall now briefly describe the abstract interface given in the specification of TCP. You should bear in mind that, although specific implementations will perform similar functions, the methods of achieving them may differ. Table 5 lists the primitives and their parameters. Most of the parameters are fairly self-explanatory but parameters of the open primitive are described below.

Table 5: TCP primitives

Primitive name	Primitive parameters
Abort	Local connection name
Close	Local connection name
Open	Local port number, foreign socket, active/passive flag, time-out, precedence, security/compartments, options Returns a local connection name
Receive	Local connection name, buffer address, byte count Returns a byte count and status of the urgent and push flags
Send	Local connection name, buffer address, byte count, push flag, urgent flag, time-out
Status	Local connection name Returns status data about the network addresses, port numbers, state, buffers, etc.

The local port number in the open primitive is the 16-bit number chosen by the application to identify itself. The foreign socket identifies the host to which it wants to be connected, and consists of the network address and port number of the destination host.

The active/passive flag indicates the type of open primitive. An active open will begin to open a specific connection immediately, whereas a passive open will cause the host to listen for an incoming connection. For example, an HTTP server may request a passive open for port 80 which indicates that it is willing to receive HTTP commands from any host that has a destination port of 80.

The open primitive returns a local connection name, which is specified in other primitives to identify the connection.

The operation of the transmission control protocol follows that of a finite state machine, and [Figure 19](#) shows a state diagram representation of the TCP connection management. However, not all possible state changes are represented in this simplified diagram. It is meant to illustrate the operation of TCP, not to be a specification.

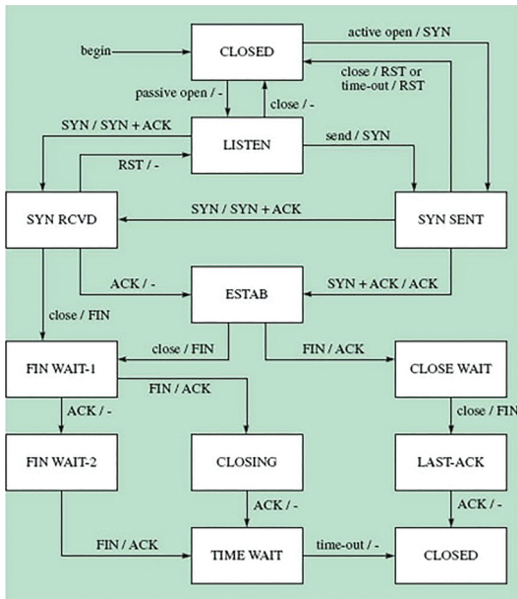


Figure 19 TCP connection finite state machine

SAQ 9

Amend the signal sequence diagram in [Figure 19](#) to show the state of each host after it receives an incoming signal. Assume that the TCP entities in both hosts start in the 'CLOSED' state. You will need to infer the arrival of a primitive for some state transitions and you should include the primitives on the diagram.

Answer

See Figure 20. The names of the states appear in rectangles at appropriate points in the signal sequence diagram.

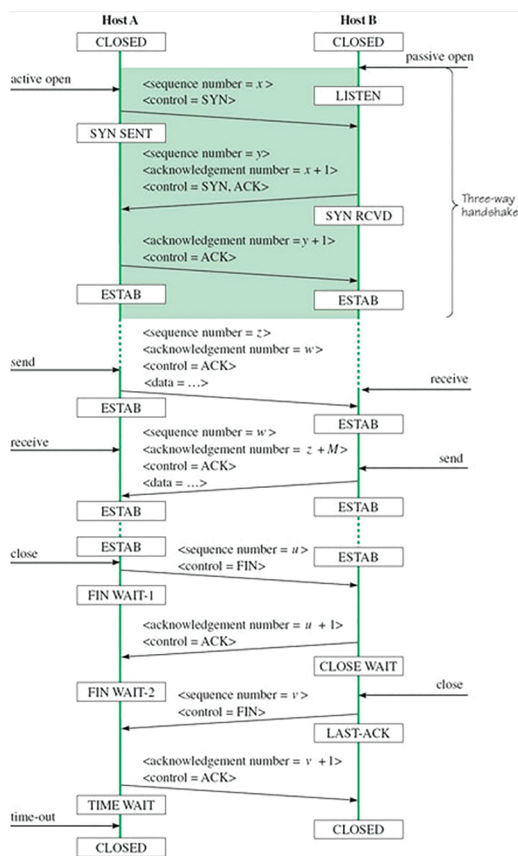


Figure 20 Answer to SAQ 9

SAQ 10

Briefly describe what you believe is the function of each primitive listed in [Table 5](#) based on its name and its parameters.

Answer

My descriptions are given below:

- **abort** – to terminate a connection and discard all pending data awaiting transmission or reception;
- **close** – to clear a connection once all data awaiting transmission is transmitted;
- **open** – to open a connection, in either passive or active mode;
- **receive** – to allocate buffer space to store incoming data;
- **send** – to indicate the buffer containing the data to be sent over a connection;
- **status** – to request information about the status of a connection.

The primitives are described more fully in RFC 793.

3.5 Internet protocol (IP)

At the time of writing (2002), two versions of IP are available: versions 4 and 6. In this section I shall describe version 4, which is abbreviated to IPv4, as it is the more widely available version. Version 6 may eventually replace version 4 because it has some additional features that may prove essential for multiservice networks.

IPv4 is the main TCP/IP protocol in the internetwork layer of the TCP/IP reference model. It supports a connectionless service between hosts in an internetwork and its principal function is to forward the protocol data units, called *datagrams* in IP terminology. This is achieved by each datagram carrying a unique address of its destination. Figure 21 shows the format of an IPv4 datagram.

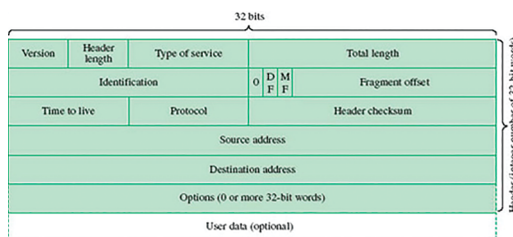


Figure 21 IPv4 datagram

The following fields are identified in Figure 21:

- version number (4 bits) – identifies the version of IP that sent this datagram;
- header length (4 bits) – contains the number of 32-bit words in the header;
- type of service (8 bits) – identifies the service requirements, such as reliable data transfer or fast data transfer, which can influence the choice of route;
- total length (16 bits) – gives the total number of bytes in the header and user data field;
- identification (16 bits) – provides a means of identifying datagrams that carry parts of the same data from the transport layer. A router may need to fragment a datagram for transfer through sections of an internetwork, and if this occurs all datagrams carrying fragments of the original datagram have the same value in this field so that they can be re-assembled before being passed to the transport layer of the destination host;
- DF (1 bit) – the ‘don’t fragment flag’ indicates that a datagram should not be fragmented;
- MF (1 bit) – the ‘more fragments flag’ indicates that a datagram is not the last fragment;
- fragment offset (13 bits) – indicates where the fragment belongs when measured in multiples of 8 bytes. The first fragment has an offset of 0;
- time to live (8 bits) – gives a way of indicating how long a datagram should exist in an internetwork. In practice, the time-to-live field is a hop counter, which is decremented by 1 each time a datagram is forwarded by a router. A datagram is discarded if the time-to-live value reaches 0;
- protocol (8 bits) – identifies the protocol that is using IP, e.g. TCP or UDP;
- header checksum (16 bits) – contains a simple form of cyclic redundancy check for a datagram, but is applied only to the header information;

- source address (32 bits) – contains the network address of the host which sent the datagram;
- destination address (32 bits) – contains the network address of the host which should receive the datagram;
- options (0 or more 32-bit words) – provides a way of identifying optional features associated with security or routing;
- user data (optional) – contains the data from the higher layer.

The IPv4 network addresses, which appear in the source and destination fields, are very important and are described in the next subsection.

3.5.1 IPv4 Addresses

The allocation of addresses on the Internet is controlled by the Internet Assigned Numbers Authority (IANA), although authority is delegated to several local registries. IPv4 addresses may be interpreted in two ways. Initially, they were divided into distinct ranges of addresses called classes, but this proved to be inflexible and now a more flexible scheme, called classless addressing, dominates IPv4 internetworks. I shall describe both ways of interpreting IPv4 addresses because the limitation of the first interpretation leads naturally to the need for the second interpretation. The formats of the four allocated classes of IPv4 addresses are shown in Figure 21 and their ranges are listed in Table 6. The values in the table follow the standard dotted-decimal convention of splitting each 32-bit address into four bytes, and expressing the values of the bytes as denary numbers separated by full stops.

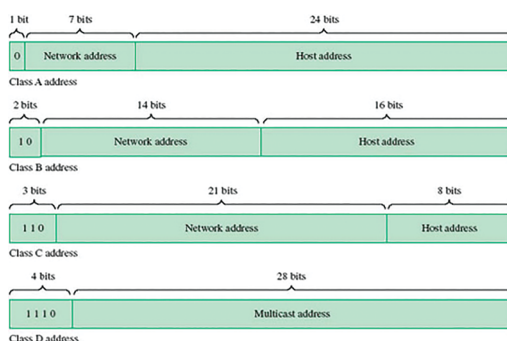


Figure 22 IPv4 Internet address formats

Table 6: IPv4 Internet address ranges

Class	Range	
	Lowest	Highest
A	0.0.0.0	127.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255

Addresses in the range 240.0.0.0 to 255.255.255.255 are reserved for future use.

IPv4 address classes A to C identify single hosts, and are called *unicast addresses*.

Looking at Figure 22, you can see that the IPv4 addresses consist of three parts: a prefix

identifying the class of address, the address of a network, and the address of a host within the network. For instance, a class C address is identified by the prefix 110, the network address consists of 21 bits, and the host address consists of eight bits. This means that, in principle, there can be $2^{21} = 2097152$ networks in an internetwork with class C addresses, and each single network can have $2^8 = 256$ hosts connected to it. In practice, some addresses are reserved and cannot be allocated to networks or hosts.

SAQ 11

To which class does the IPv4 address 157.107.34.5 belong?

Answer

The most direct approach to this question is by reference to Table 6. The IPv4 address 157.107.34.5 is within the address range for class B. You could also have answered this question by expressing the first byte of the address as a binary number and then identifying the prefix. In this case 157 is equal to the binary number 10011101 which is a class B address since the first two bits are 10.

Class D addresses are allocated to groups of hosts rather than single hosts. The group is identified by a single 28-bit address, called a *multicast address*, and the datagram is delivered to all the hosts in the group. Some multicast addresses are reserved for specific groups of hosts, such as all routers connected to a LAN. Other multicast groups can be constructed by hosts sending appropriate management messages. Multicast datagrams are routed through an internetwork by special multicasting routers.

The main benefit of multicasting is that routers can optimise the distribution of datagrams to avoid unnecessary duplication. For instance, if a host wishes to send a single datagram to 100 destinations by unicasting, it has to send 100 copies of the datagram. However, if all the 100 destinations belong to a single multicast group, the originating host can send a single multicast datagram and the routers will duplicate the datagram only where necessary to deliver it to destinations with which they have different interfaces. This reduces the total number of copies of the datagram that exist in the internetwork.

The division of IPv4 unicast addresses into three classes is not a very efficient way of allocating addresses. A network with, say, 300 hosts would have to be given a class B address, which allocates 16 bits to host addresses and therefore can accommodate $2^{16} = 65536$ hosts. However, only 300 of these addresses would be allocated – the rest would be wasted. A classless addressing scheme has therefore been introduced to reduce the inefficiency in the original IPv4 addressing scheme.

Classless IPv4 addresses are written in the same dotted-decimal notation as the original addressing scheme, but are followed by a forward slash and the number of bits in the network part of the address, which precedes the host address. For example, the classless address 196.19.0.48/16 identifies the network address as 196.19 (11000100 00010011) and the host address within the network as 0.48 (00000000 00110000).

The routing of IPv4 datagrams is based on the network address component of the IPv4 destination address. The network address can be conveniently isolated by using the simple Boolean operator AND with a binary number which has ones in all bit positions of the network address component. This process is called *masking* and the binary number used to isolate the network address component is called a bit mask. For example, the network address in the IP address 196.19.0.48/16 can be isolated by the bit mask 255.255.0.0. The bit masks are set up in each router as part of the routing functions of an internetwork.

The unit team hope you will find spreadsheets very useful during your study of this unit. The following activity and SAQ ask you to use fairly simple spreadsheets.

The worksheet 'IPv4 classes' in the spreadsheet 'Protocols' converts IPv4 addresses expressed as 32-bit binary numbers to dotted-decimal numbers. The spreadsheet supplied below is set up to convert the upper and lower addresses in the four classes of addresses. Use this spreadsheet to confirm the address ranges in [Table 6](#).

Some calculators, such as the Windows calculator, convert between binary and denary numbers, and you may like to check the answers in the spreadsheet. If you are unfamiliar with binary numbers you may find it worthwhile using this spreadsheet to convert other binary numbers.

Click on the 'View document' link below to open the spreadsheet. You will find it attached to a PDF.

[View document](#)

SAQ 12

Find the bit masks and hence the network addresses for the following IPv4 classless addresses:

- (a) 197.134.5.45/12
- (b) 220.5.47.101/10
- (c) 175.121.56.221/8

The worksheet '[SAQ 6](#)' The worksheet SAQ 3.3 in the spreadsheet. 'Protocols' may be used for this SAQ. You may like to change the values in this spreadsheet to see the effects of other bit masks.

Answer

The bit masks and network addresses are given below. Various short cuts are possible but, going through all the steps, the network address is isolated by first converting the IP address to a binary number and then ANDing with the mask.

(a) The bit mask has 1s in the first 12 bit positions:

11111111 11110000 00000000 00000000

which is 255.240.0.0 in the dotted-decimal form.

The network address is isolated as follows:

11000101	10000110	00000101	00101101	(197.134.5.45)
11111111	11110000	00000000	00000000	(255.240.0.0)
11000101	10000000	00000000	00000000	(197.128.0.0)

The network address component is 110001011000.

(b) The bit mask has 1s in the first 10 bit positions:

11111111 11000000 00000000 00000000 (255.192.0.0)

The network address is isolated as follows:

11011100	00000101	00101111	01100101	(220.5.47.101)
11111111	11000000	00000000	00000000	(255.192.0.0)
11011100	00000000	00000000	00000000	(220.0.0.0)

The network address component is 1101110000.

(c) The bit mask has 1s in the first eight bit positions:

11111111 00000000 00000000 00000000

The network address is isolated as follows:

10101111	01111001	00111000	11011101	(175.121.56.221)
11111111	00000000	00000000	00000000	(255.0.0.0)
10101111	00000000	00000000	00000000	(175.0.0.0)

The network address component is 10101111.

3.5.2 IP forwarding

At the internetwork layer, forwarding is based on the network address part of the IPv4 address. Two basic forms of forwarding are carried out by a router or host – direct and indirect. *Direct forwarding* takes place where the destination IPv4 address is on a network attached to the router or host. *Indirect forwarding* takes place where the destination IPv4 address is not on a network connected to the router or host, and the datagram therefore has to be forwarded to a router further along the path through the internetwork. Where a router is connected to several networks, forwarding is more complicated than in the case of hosts, which are typically connected to a single network. Figure 23 shows a simple internetwork in which we can look at examples of direct and indirect forwarding.

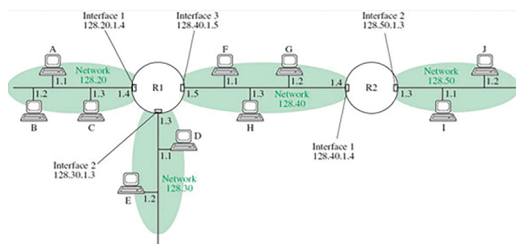


Figure 23 Direct and indirect forwarding

In Figure 23 there are four networks forming part of an internetwork. The IPv4 addresses are class B and the network addresses are given in the figure. Each of hosts A to J has a connection to only one network and their IP addresses can be found by prefixing the network addresses to the host addresses shown in the figure. For instance, the full address for host B is 128.20.1.2 and that for host G is 128.40.1.2. The router labelled R1 is connected to three networks, whereas router R2 is connected to only two. In each case the interface to each network has a separate IPv4 address.

Suppose that host B sends a datagram to host C. Assume that host B knows the IP address of host C (128.20.1.3) and from this it knows that C is on the same network as itself. This means that B can invoke the services of its host-to-network layer to deliver the datagram.

Now suppose that host A wishes to send a datagram to host J. Assume that A knows the IPv4 address of the destination host J (128.50.1.2) but this time from the address it knows that it does not have a direct connection to J. Host A therefore has to forward the datagram to a router. The actual delivery of the datagram is one of the functions performed by the host-to-network layer. The forwarding information is stored in a data structure called a *forwarding table*. In this example the forwarding table for host A would

be very simple, as is shown in Table 7. Note that the destination address (J) in the IP datagram is not changed.

Table 7: Forwarding table for host A

Network address	Direct/Indirect	Router	Interface number
128.20	Direct	–	1
128.30	Indirect	128.20.1.4	1
128.40	Indirect	128.20.1.4	1
128.50	Indirect	128.20.1.4	1

Host A forwards datagrams directly if they are addressed to network 128.20, or indirectly via router R1 if they are addressed to any of the other networks.

Table 8 shows the forwarding table for router R1. In this case, R1 has direct connections to three networks (128.20, 128.30, 128.40) and an indirect connection, via router R2, to the fourth network (128.50).

Table 8: Forwarding table for router R1

Network address	Direct/Indirect	Router	Interface number
128.20	Direct	–	1
128.30	Direct	–	2
128.40	Direct	–	3
128.50	Indirect	128.40.1.4	3

To give you some idea of scale, some routers in the Internet have forwarding tables containing of the order of 50000 entries. In practice, hosts and routers do not have entries for all network addresses and a default router may be specified for those network addresses that are not recognised. In addition, several network addresses that have the same interface may be amalgamated into a single entry by a process called route aggregation. This is discussed in more detail in the next section under the topic of forwarding equivalence classes. A datagram is discarded if it has a destination network address for which no entry appears in the forwarding table and if there is no default entry in the forwarding table.

For the host-to-network layer to deliver the datagram, the host must know the physical address of the destination host. The physical address is not the same as the IP address because different networks have different addressing schemes for identifying systems. For instance, terminals connected to an Ethernet LAN are given addresses by the manufacturer and these addresses do not change if the terminals are transferred to different LANs. How a host gets the physical address on a LAN is the topic of the next section. Other protocols exist for other types of network.

3.5.3 Address resolution on LANS

A host finds the hardware address that corresponds to an IP address by referring to a look-up table that contains the translation between the two types of address. The look-up table could be maintained manually by a network administrator, but this would be very slow. A far better solution is for the host to maintain the table automatically. Whenever a

host has to send a datagram to an unknown IP address it sends a message to all the other hosts on the LAN. The host that recognises the IP address as its own will respond by returning a message with its physical address. The original host will then pass the physical address and datagram to its host-to-network layer for delivery to the destination host. The translation between IP address and physical address is stored in the look-up table so that, should another datagram arrive for delivery to the same destination host, the original host only has to refer to the look-up table. The protocol that specifies the communication between hosts for the translation of addresses is called the *address resolution protocol* (ARP).

3.5.4 Primitives

The two abstract primitives described in the specification of IPv4 are given in Table 9 but, as with TCP, different systems may implement the interface in different ways. In some computer systems the TCP and IP software may be written as a single software module, and in these cases there is not a clear interface between the two layers.

Table 9: IPv4 primitives

Primitive name	Primitive parameters
Recv	Buffer pointer, protocol, source address, destination address, type of service, length of buffer, optional user data Returns a report on the outcome of the primitive
Send	Source address, destination address, protocol, type of service, time to live, buffer pointer, length of buffer, identifier, don't fragment flag, optional user data Returns a report on the outcome of the primitive

You should be able to match most of the parameters to fields in IPv4 datagrams.

SAQ 13

Identify the protocol data units and primitives required to send an HTTP command between a client and server in the transport and internetwork layers.

Answer

Assuming that a connection does not already exist between the HTTP application running on the client and the HTTP server application, the client application layer requests that the transport layer sets up a connection by issuing a TCP open primitive. Note that the foreign socket includes port number 80. (Before this request, the application would have to translate the host name to a network address to create the foreign socket value required by the open primitive.)

The transport layer encapsulates the TCP segment that is setting up the connection in an IP datagram by invoking an IP send primitive. In other words, the TCP segment is the optional user data parameter of an IP send primitive. The TCP segment that is the third part of the three-way handshake, acknowledging the connection in the opposite direction, is encapsulated in another IP datagram. This is accomplished by the transport layer invoking another IP send primitive.

The HTTP client application requests that the data representing the HTTP command be transferred to the HTTP server application by invoking a TCP send primitive. The transport layer transfers this data by invoking an IP send primitive, thereby encapsulating the TCP segment in an IP datagram. Depending on the size of the command, the transport layer may have to fragment the user data over several segments. Similarly, during the transport of the datagram(s) further fragmentation may take place.

4 ATM protocol architecture

4.1 What is ATM protocol architecture?

The *asynchronous transfer mode* (ATM) protocol architecture is designed to support the transfer of data with a range of guarantees for quality of service. The user data is divided into small, fixed-length packets, called cells, and transported over virtual connections. ATM operates over high data rate physical circuits, and the simple structure of ATM cells allows switching to be performed in hardware, which improves the speed and efficiency of ATM switches.

Figure 24 shows the reference model for ATM. The first thing to notice is that, as well as layers, the model has planes. The functions for transferring user data are located in the user plane; the functions associated with the control of connections are located in the control plane; and the co-ordination functions associated with the layers and planes are located in the management planes.

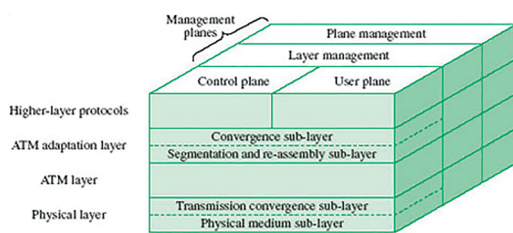


Figure 24 ATM reference model

The three-dimensional representation of the ATM protocol architecture is intended to portray the relationship between the different types of protocol. The horizontal layers indicate the encapsulation of protocols through levels of abstraction as one layer is built on top of another, whereas the vertical planes indicate the functions that require co-ordination of the actions taken by different layers. An advantage of dividing the functions into control and user planes is that it introduces a degree of independence in the definition of the functions: the protocols for transferring user data (user plane) are separated from the protocols for controlling connections (control plane).

The protocols in the ATM layer provide communication between ATM switches while the protocols in the ATM adaptation layer (AAL) operate end-to-end between users. This is illustrated in the example ATM network in Figure 25.

Two types of interface are identified in Figure 24: one between the users and the network (user-network interface), and the other between the nodes (switches) within the network (network-node interface).

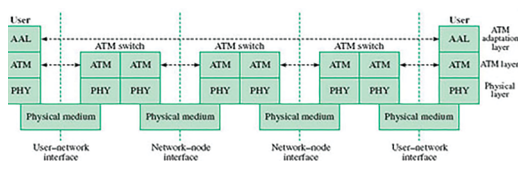


Figure 25 ATM network

Before describing the functions of the three layers in the ATM reference model, I shall briefly describe the format of ATM cells. Figure 26 shows the two basic types of cell.

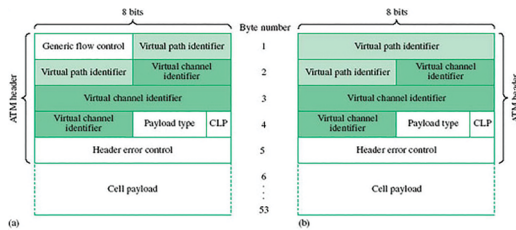


Figure 26 (a) ATM cells at the user-network interface; (b) ATM cells at the network-node interface

Each ATM cell consists of 53 bytes: the header is five bytes long and the remaining 48 bytes (the cell payload) carry information from higher layers. The only difference between the two types of ATM cell is that the cells at the user-network interface carry a data field for the flow control of data from users. This means that only eight bits are available for virtual path identifiers, rather than 12 bits at the network-node interface.

The virtual connections set up in ATM networks are identified by the combination of the *virtual path identifier* and *virtual channel identifier* fields shown in Figure 26. These two fields provide a hierarchy in the numbering of virtual connections, whereby a virtual path contains a number of virtual channels as is illustrated in Figure 27. An advantage of this hierarchy is that in some cases the switching of ATM cells may be based on the virtual path identifier alone.

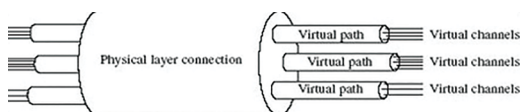


Figure 27: Virtual path and virtual channel relationship (Source: adapted from ITU-T 1–150, 1999, Figure 1)

SAQ 14

how many different virtual connections can exist between a user and an ATM network, and between a node and an ATM network?

Answer

between a user and an ATM network a total of 24 bits are available for identifying virtual connections; therefore there can be $2^{24} = 16777216$ connections. At the network-node interface a total of 28 bits are available, so the number of connections possible is $2^{28} = 268435456$.

The *payload type* field identifies the type of cell. I do not intend to describe the specific types of ATM cell, but there are types for carrying user information, signalling information for controlling virtual connections, and management information. There are two basic types of user information cell: one in which congestion has been identified and one in which it has not.

The *cell loss priority* (CLP) field is a single bit; if the bit is 0 that cell has a high priority, and if the bit is 1 the cell has a low priority. This information may influence the decision whether to discard cells if a network becomes congested.

The *header error control* field contains a cyclic redundancy check on the other bytes in the header.

4.2 ATM layers

In this section I shall briefly review some of the main functions of the ATM layers but I shall not go into too much detail because at this stage we are interested in only the general points about protocols.

4.2.1 ATM physical layer

The *ATM physical layer* is divided into two sub-layers: the transmission convergence sub-layer and the physical medium sub-layer.

Functions of the *transmission convergence sub-layer* include generating and receiving cells, and generating and verifying the cyclic redundancy check in the header error control field. For correct interpretation of ATM cells it is important to identify the beginning of a cell. In theory, if ATM cells are transmitted as a continuous stream of bits, once a receiver has found the start of one cell, the start of the next cell starts $53 \times 8 = 424$ bits later. However, this still leaves the problem of identifying the start of the first ATM cell. The sending device inserts a cyclic redundancy check word in the header error control field in each ATM cell. A receiving device performs cyclic redundancy checks on 40 consecutive bits (five bytes) of the bit stream, so valid headers will pass this test. Of course, it is possible that any 40 bits may pass a cyclic redundancy check by chance. To avoid the possibility of misinterpretation, a receiver will check the header error control bytes of the next few cells before assuming that it has managed to synchronise with the arrival of ATM cells over a link. Unfortunately, because of mismatches in the timing of senders and receivers, it is possible to lose bits. Therefore, the receiver should still monitor the header error control field. If several consecutive cells fail the cyclic redundancy check, the receiver assumes that it has lost synchronisation over that link and starts again to look for ATM headers by examining 40 consecutive bits. This process is shown as a state diagram in Figure 28. The number of correct header error control checks that should occur before the receiver assumes synchronisation is expressed as δ° and the number of incorrect checks that can occur before it assumes loss of synchronisation is expressed as α° .

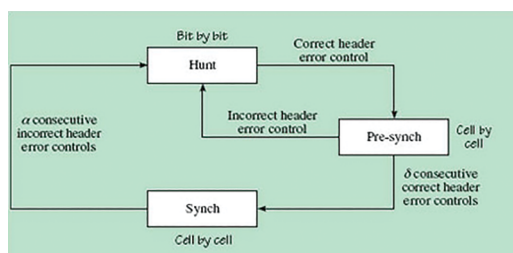


Figure 28 ATM cell synchronisation

SAQ 15

Why is it not assumed that synchronisation is lost after a single header error control check failure, i.e. $\alpha^\circ = 1$?

Answer

Header error control check failures are more likely to be caused by simple transmission errors of misinterpreting a 1 for a 0, or vice versa, rather than a timing error causing a loss or insertion of a bit. Therefore, it is sensible to discard the ATM cell that failed the check, but still assume for the time being that the link is synchronised.

The functions of the ATM *physical medium sub-layer* are associated with the transmission of bits over a specific physical medium. The main transmission m

4.2.2 ATM layer

The primary functions of the *ATM layer* are associated with the routing and switching of ATM cells. Because ATM cells are packets, the switches are packet switches and the switching operation can be called forwarding, but by convention, because the ATM layer provides a connection-oriented service, the term 'forwarding' is generally not used.

The path cells take and the resources allocated to them depend on their service category. This is determined when a virtual connection is established. The following service categories are recognised by the ATM layer:

- Constant bit rate – a constant data rate is allocated and is continuously available for the duration of a connection.
- Real-time variable bit rate – there are some commitments about the data rate to be made available, and the delay and variation in delay are tightly controlled.
- Non-real-time variable bit rate – there are some commitments about the data rate to be made available, but no delay limits are placed on the delivery of cells.
- Unspecified bit rate – there are no commitments about the data rate to be made available.
- Available bit rate – the data rate made available may be changed during the time a connection is maintained.
- Guaranteed frame rate – there is a commitment about the minimum data rate of a connection.

There are two types of virtual circuit – switched virtual circuits and permanent virtual circuits. The two types are similar in that they must be established before user data can be transferred; the difference is how they are set up. *Switched virtual circuits* are set up in response to user requests to transfer data and are released once that exchange has been completed. *Permanent virtual circuits* are set up by management activities in response to contracts established between users and are expected to last much longer than switched virtual circuits. The word 'permanent' may be misleading because permanent virtual circuits do change in a network, but they change relatively infrequently, and from the point of view of users they are always available. Both types of virtual circuit are controlled by functions in the control plane of the ATM reference model and are very important for the routing and switching of ATM cells.

I do not intend to go into the details of particular control protocols adopted by ATM. For our purposes here it is sufficient to understand that virtual circuits are established in response to set-up messages which contain the address of the destination. These connections are virtual connections, similar to TCP connections, but they take place at a lower level of abstraction. ATM switches examine the destination address in a set-up message and decide the best path to take for the service category intended for that connection. Each link in the path is identified by a virtual path identifier and a virtual channel identifier. Once

a virtual circuit has been established, ATM cells carrying user data are switched according to their virtual path and virtual channel identifiers. For the purposes of switching, permanent virtual circuits are treated identically to switched virtual circuits.

The switching information in each ATM switch takes the form of a forwarding table, like the example in Table 10. For connection-oriented networks, a forwarding table is sometimes called a connection table and the term ‘forwarding’ is restricted to connectionless networks.

Table 10: ATM forwarding table

Incoming			Outgoing		
Interface number	Virtual path identifier	Virtual channel identifier	Interface number	Virtual path identifier	Virtual channel identifier
1	1	5	2	3	5
1	2	4	4	2	7
...	...	...	...	...	...

4.2.3 ATM adaptation layer

The basic function of the *ATM adaptation layer* is to convert the user data supplied by higher layers into 48-byte blocks of data. The ATM adaptation layer is divided into two sub-layers – the convergence sub-layer, and the segmentation and re-assembly sub-layer. The *convergence sub-layer* provides services to higher layers through a set of protocols, but I do not need to describe these here. The *segmentation and re-assembly sub-layer* separates the messages from the convergence sub-layer into ATM cells. Each of the two sub-layers adds some protocol information, which is transported in the payload of ATM cells as illustrated in Figure 29.

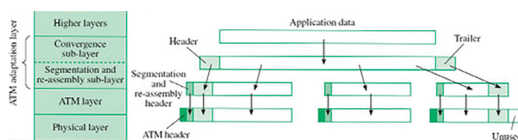


Figure 29 ATM adaptation layer functions (Source: based on Tanenbaum, 1996, Figure 6.38)

4.3 IP over ATM

You have seen that the IP protocol supports a connectionless service, and the ATM and TCP protocols support a connection-oriented service.

SAQ 8 (Revision)

What are the relative advantages and disadvantages of connectionless and connection-oriented services for transporting data through a communication network?

Answer

My list of the advantages and disadvantages of connectionless and connection-oriented services is given in Table 11. The list is not exhaustive and you may have included some other factors. Also you may wish to qualify some of those I have listed.

Table 11: Answer to SAQ 8 (Revision)

Type of service	Advantages	Disadvantages
Connectionless	Data can be sent immediately without the overhead of setting up a connection	No intrinsic flow control No intrinsic error correction
	It is robust against link failures	Forwarding is based on full destination address
	The network switches do not need to maintain connections	Order of packets is not guaranteed There can be wide variation in the delay
Connection-oriented	Flow control can be implemented within the service	Overhead of setting up a connection before data is sent
	Error control can be implemented within the service	Switches have to maintain connections
	Order of packets is maintained	
	Forwarding is done according to simple connection identifiers	

The Internet and internetworks in general consist of many physical networks, which may be of different types. After all, one of the main advantages of the TCP/IP protocol suite is its compatibility with many different network technologies. The host-to-network layer in the TCP/IP protocol model may transfer data over a range of network links – relatively slow modem links to homes, fast LANs interconnecting many computers within a building, fast private point-to-point links rented from telephone network operators. The example I wish to explore in this section is sending IP traffic over an ATM network.

Telephone network operators have installed a large amount of ATM equipment and it is likely that in the future a significant amount of traffic will be transported over ATM networks for part of its path between users. The ATM portions of the paths are of no interest to users because all they are concerned about is the quality of service they receive, not how it is achieved.

Several techniques exist for sending IP traffic over ATM networks. The technique I shall describe is label switching. However, you should bear in mind that there are other approaches and that label switching has wider application than supporting IP traffic over ATM networks. I shall ignore many of the intricate problems that arise when two significantly different technologies work together.

Before discussing label switching, however, I need to review the two basic mechanisms for transporting packets in packet-switched networks, which I shall call forwarding and routing. Unfortunately, these terms tend to be used interchangeably in the literature and their exact meaning has to be derived from the context. In this section I shall try to be more precise and define *forwarding* as the process of receiving a packet and passing it on to the next stage in the path to its destination, that is, packet switching. This generally

involves a packet switch referring to a table that identifies the next path for each destination. This table is commonly called a forwarding table (see examples discussed earlier for ATM and IP), but it is also frequently called a routing table.

The entries in a forwarding table are determined by the *routing* process. The best path through a network is chosen from information about which links are available between switches. Selection of the best path can be based on the current level of traffic, which means that the contents of a forwarding table will change. Changes are made in the background to the forwarding process. Protocols that are specific to a network transport routing information between switches.

Generally in a large IP internetwork, it is impracticable for each router to contain an entry for each network address, so several network addresses are amalgamated into a single entry called a *forwarding equivalence class*. For example, imagine that a router knows that all packets destined for network addresses 132.141.0.0/16 (10000100 10001101), 132.152.0.0/16 (10000100 10011000) and 132.172.0.0/16 (10000100 10101100) should be forwarded over the same path. The 10 most significant bits of the 16-bit network addresses have the same value for all three addresses, so the forwarding table could have a single entry for the value 132.128.0.0/10 (10000100 10000000).

SAQ 16

What is the longest forwarding equivalence class address if datagrams for the network addresses 201.192.0.0/8, 200.128.0.0/8 and 201.64.0.0/8 are forwarded over the same interface?

Answer

Converting the network addresses to binary values:

201	11001001
200	11001000
201	11001001

The longest match is on the first seven bits of the network addresses, which gives 200/7 or 201/7. The network addresses 200/7 and 201/7 are indistinguishable, so either value could be chosen.

With forwarding equivalence classes it is possible for a network address to match more than one entry in a forwarding table and the path chosen will be the one with the *longest matching address*.

SAQ 17

The following network addresses appear in a forwarding table:

- 182.0.0.0/8
- 182.64.0.0/10
- 182.128.0.0/10
- 182.160.0.0/12
- 182.144.0.0/12.

Which of the entries match the IPv4 address 182.145.57.3?

Answer

The IPv4 address as a binary number is 10110110 10010001 00111001 00000011 and comparing this address with the table entries gives:

	Table entry	Destination network address	Result
(a)	10110110	10110110	Yes
(b)	10110110 01	10110110 10	No
(c)	10110110 10	10110110 10	Yes
(d)	10110110 1010	10110110 1001	No
(e)	10110110 1001	10110110 1001	Yes

The searching for the longest match in a forwarding table takes place in each router in a connectionless network. The forwarding decision in a connection-oriented network is based on virtual path identifiers, which tend to be of fixed lengths. This simplifies the searching process. *Label switching* in essence brings this advantage of connection-oriented networks to connectionless protocols. Within a label-switched network, packets are forwarded according to labels added to them rather than according to the destination address stored in the packet header.

A router that is located around the periphery, or boundary, of a network is called an *edge router*. At an edge router of a label-switched network the destination addresses have to be converted to labels, but once within a label-switched network forwarding is achieved by inspection of labels alone. In ATM networks the labels can be inserted in the existing fields for virtual path and/or virtual channel identifiers. In other networks it may be necessary to encapsulate the IP datagram in another packet which contains a label in its header.

SAQ 18

Label switching seems to add extra overheads to forwarding packets. Suggest what advantages it may have.

Answer

Possible advantages include the following:

- if the label-switched network extends for most of the path between end users, then most of the packet forwarding would be done more efficiently than with conventional routers;
- label switching allows IP traffic to be carried by connection-oriented networks such as ATM;
- it may be possible to incorporate greater control over the quality of service by assigning packets associated with particular applications to higher-quality virtual circuits.

Figure 30 shows the operation of IP over ATM using label switching. Although the figure shows only the entries for the paths from IP network 131.30.xx, it is a fairly complicated diagram, so you will need to spend some time working through the forwarding tables. The association of labels with IP forwarding equivalence class addresses is part of the routing process and is accomplished by a label distribution protocol in conjunction with the ATM signalling protocol for establishing virtual circuits.

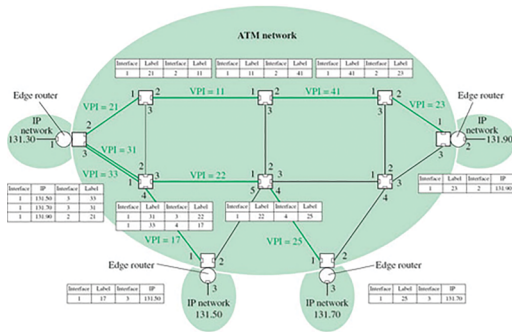


Figure 30 P over ATM

SAQ 19

Compare the functions of forwarding IP datagrams and ATM data cells.

Answer

IP datagrams are forwarded by routers (packet switches) according to network addresses and as a connectionless service. The path taken depends on the content of forwarding tables, which reflect the outcome of routing operations. The choice of path may vary during a flow of datagrams between users if the forwarding tables are updated.

ATM data cells are forwarded (switched) according to the values of their virtual path and virtual channel identifiers. These identifiers refer to either a permanent virtual path or a switched virtual path, but in both cases a connection-oriented service is provided to users.

The values of virtual path and virtual channel identifiers may vary as cells are forwarded (i.e. they have local significance), whereas network destination addresses in IP datagrams do not vary (i.e. they have global significance).

Conclusion

Now that you have completed this unit it is a good idea to reflect on what you have learned, and a good way of reflecting on your learning is to write a summary of the material. A brief summary is provided below, but this is no substitute for your own summary. You may also find it useful to note any points you are unsure of and wish to follow up by further reading. Posting a message on the OpenLearn forums board may elicit suggestions and support from fellow students.

There are two basic types of switching – circuit switching and packet switching. A physical communication channel is established for the exclusive use of a call in a circuit-switched network. In contrast, physical communication channels are shared in a packet-switched network. Packet switching is described as store-and-forward because a switch will store packets and then forward them when resources become available.

Networks can provide connection-oriented and connectionless services. In a connection-oriented service, a connection must be established before user data is transferred, whereas this is not necessary for a connectionless service. The connections need not be physical channels, and generally exist to organise the flow of packets rather than to reserve resources.

The OSI reference model is an approach to the separation of the communication functions of systems into seven layers: application, presentation, session, transport, network, data link and physical. The concept of layering is common with most protocols, although the seven layers advocated by OSI are not always followed. The vertical flow of data is represented by primitives being issued and received across layer boundaries. The four basic primitive types are: request, indication, response, and confirm. The flow of data between peer layers in different systems is called horizontal communication and is governed by protocols, which enable a layer to provide services to the layer above. One or more protocols is available to each layer. A protocol defines the structure and the meaning of protocol data units and therefore how the systems communicate. A protocol data unit in a higher layer is encapsulated in a protocol data unit in a lower layer.

Each layer, except the lowest, works at a level of abstraction based on the services provided by lower layers. For instance, higher layers can ignore the problems that arise with specific physical media, such as optical fibre and twisted pair, because the physical layer deals with problems associated with the types of physical medium.

The Internet is based on the TCP/IP protocol suite, which generally has a four-layer model: application, transport, internetwork, and host-to-network. The hypertext transfer protocol (HTTP) is an example of an application layer protocol, but there are several others. The principal transport layer protocol is the transmission control protocol (TCP), which provides a connection-oriented service. A three-way handshake provides a reliable means of establishing a connection. Sequence numbers allow any missing data to be identified. The main internetwork layer protocol is the internet protocol (IP), which provides a connectionless service. IPv4 addresses identify all hosts on an internetwork and have 32 bits. The number of bits allocated to host and network addresses is determined either by the uppermost bits if the class form of addressing is used, or by masks if the classless form of addressing is used. The forwarding of IP datagrams is accomplished by forwarding tables, which are set up by the routing process of an internetwork. Entries in a forwarding table denote either a direct path or an indirect path

for a network address. In the former case the router is able to forward the datagram over a direct connection. In the latter case the final destination is on a network to which the router does not have a direct connection.

The asynchronous transfer mode (ATM) protocol is based on 53-byte cells. The ATM layer provides a connection-oriented service and several service categories are available: constant bit rate, real-time variable bit rate, non-realtime variable bit rate, unspecified bit rate, available bit rate and guaranteed frame rate. A virtual connection is identified by the combination of a virtual path identifier and a virtual channel identifier.

One approach to providing IP over an ATM network is label switching. Edge routers of an ATM network convert between forwarding equivalence classes (i.e. significant parts of IP network addresses) and labels. The labels are transferred in the virtual path identifier and/or virtual channel identifier fields in ATM headers.

SAQ 20

Where possible, give examples and brief descriptions from TCP/IP and ATM protocols of the following OSI general functions identified in Section 2:

- connection control
- data flow
- segmentation and re-assembly
- sequencing
- acknowledgement
- error control
- synchronisation
- addressing and routing
- multiplexing and demultiplexing.

This unit has not explicitly addressed all the above issues, so you may not be able to give examples of every function.

Answer

My examples are in Table 12, but you may have given others.

Table 14: Answer to SAQ 20

Layer function	TCP/IP example	ATM example
Connection control	The establishment and clearing of TCP connections	The establishment and clearing of switched and permanent virtual paths
Data flow	The window size, sequence and acknowledgement fields in TCP protocol data units may be used to control the flow of data, but this is not discussed	No examples are given, but control of the flow of data is an important part of ATM
Segmentation and re-assembly	Segmentation and re-assembly may involve both the TCP and IP protocols. The stream of bytes	The ATM adaptation layer segments the user data into 48-byte blocks of data. The ATM

	received by the transport layer is segmented into TCP segments, and these may be further fragmented at the internetwork layer. The appropriate layer at the receiving system re-assembles the original data units	adaptation layer in the receiving system re-assembles the original data units
Sequencing	Sequence numbers are available in the TCP segments	This is not discussed, but sequencing may take place at the ATM adaptation layer
Acknowledgement	Acknowledgement numbers are available in TCP segments	This is not discussed, but acknowledgement may take place at the ATM adaptation layer
Error control	There are checksum fields in TCP and IP protocol data units, which imply that there may be some error detection available, but this is not discussed	There is a header error control field in ATM cells, so some form of error detection is available, but this is discussed only in the context of cell synchronisation
Synchronisation	Synchronisation of TCP segment sequence numbers takes place when connections are established	ATM cell synchronisation takes place at the ATM physical layer
Addressing and routing	Both the TCP and IP protocols use addresses. Network addresses are used at the internetwork layer for routing datagrams. Port addresses are used at the transport layer for identifying TCP connections	Network addresses and routing are not discussed in the context of ATM, but virtual connections are identified by virtual path and virtual channel identifiers
Multiplexing and demultiplexing	Multiplexing and demultiplexing of TCP connections are permitted through port numbers	Virtual path and virtual channel identifiers implement multiplexing and demultiplexing by allowing several virtual connections to be supported by a single physical transmission medium

Keep on learning



Study another free course

There are more than **800 courses on OpenLearn** for you to choose from on a range of subjects.

Find out more about all our [free courses](#).

Take your studies further

Find out more about studying with The Open University by [visiting our online prospectus](#).

If you are new to university study, you may be interested in our [Access Courses](#) or [Certificates](#).

What's new from OpenLearn?

[Sign up to our newsletter](#) or view a sample.

For reference, full URLs to pages listed above:

OpenLearn – www.open.edu/openlearn/free-courses

Visiting our online prospectus – www.open.ac.uk/courses

Access Courses – www.open.ac.uk/courses/do-it/access

Certificates – www.open.ac.uk/courses/certificates-he

Newsletter –

www.open.edu/openlearn/about-openlearn/subscribe-the-openlearn-newsletter

References

Coffman, K. G. and Odlyzko, A. (1998) 'The size and growth rate of the Internet', *First Monday*, Vol. 3, Issue 10, <http://firstmonday.org>

- ITU-T 1–150 (1999) *B-ISDN Asynchronous Transfer Mode Functional Characteristics*, ITU-T.
- ITU-T X.200 (1994) *Open Systems Interconnection – Model and Notation*, ITU-T. (Also known as ISO/IEC 7498–1.)
- RFC 793 (1981) *Transmission Control Protocol*, Defense Advanced Research Project Agency.
- RFC 2616 (1999) *Hypertext Transfer Protocol – HTTP/1.1*, Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.
- Tanenbaum, A. S. (1996) *Computer Networks*, 3rd edn, Prentice Hall.
- Further reading**
- AF-TM-121.000 (1999) *ATM Traffic Management Specification*, Version 4.1, ATM Forum.
- Comer, D. E. (2000) *Internetworking with TCP/IP. Vol. 1, Principles, Protocols and Architecture*, 4th edn, Prentice Hall.
- Davie, B., Doolan, P. and Rekhter, Y. (1998) *Switching in IP Networks: IP Switching, Tag Switching and Related Technologies*, Morgan Kaufmann.
- Haubs, G. (2000) 'Internet offloading today, voice offloading tomorrow', *Telephony*, Vol. 238, No. 2, pp. 54–8.
- ISO 8648 (1988) *Open Systems Interconnection – Internal Organization of the Network Layer*, ISO.
- ITU-T X.210 (1993) *Conventions for the Definition of OSI Services*, ITU-T.
- Keshav, S. (1997) *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, Addison Wesley.
- Peterson, L. L. and Davie, B. S. (1999) *Computer Networks: A Systems Approach*, 2nd edn, Morgan Kaufmann.
- RFC 791 (1981) *Internet Protocol*, Defense Advanced Research Project Agency.
- RFC 792 (1981) *Internet Control Message Protocol*, Postel, J.
- RFC 959 (1985) *File Transfer Protocol*, Postel, J. and Reynolds, J.
- RFC 1034 (1987) *Domain Names – Concepts and Facilities*, Mockapetris, P.
- RFC 1035 (1987) *Domain Names – Implementation and Specification*, Mockapetris, P.
- RFC 1112 (1989) *Host Extensions for IP Multicasting*. Deering, S. E.
- RFC 1122 (1989) *Requirements for Internet Hosts – Communication Layers*, Internet Engineering Task Force.
- RFC 1123 (1989) *Requirements for Internet Hosts – Application and Support*, Internet Engineering Task Force.
- RFC 1180 (1991) *A TCP/IP Tutorial*, Socolofsky, T. and Kale, C.
- RFC 1483 (1993) *Multiprotocol Encapsulation over ATM Adaptation Layer 5*, Heinanen, J.
- RFC 1518 (1993) *An Architecture for IP Address Allocation with CIDR*, Rekhter, Y. and Li, T.
- RFC 1519 (1993) *Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy*, Fuller, V., Li, T., Yu, J. and Varadham, K.
- RFC 1577 (1994) *Classical IP and ARP over ATM*, Laubach, M. RFC 1700 (1994) *Assigned Numbers*, Reynolds, J. and Postel, J.
- RFC 1737 (1994) *Functional Requirements for Uniform Resource Names*, Sollins, K. and Masinter, L.

RFC 1738 (1994) *Uniform Resource Locators (URL)*, Berners-Lee, T., Masinter, L. and McCahill, M.

RFC 1889 (1996) *RTP: A Transport Protocol for Real-Time Application*, Schulzine, H., Casner, S., Frederick, R. and Jacobson, V.

RFC 2050 (1996) *Internet Registry IP Allocation Guidelines*, Hubbard, K., Koster, M., Conrad, D., Karrenberg, D. and Postel, J.

RFC 2098 (1997) *Toshiba's Router Architecture Extension for ATM: Overview*, Katsube, Y., Nagami, K. and Esaki, H.

RFC 2101 (1997) *IPv4 Address Behaviour Today*, Carpenter, B., Crowcroft, J. and Rekhter, Y.

RFC 2105 (1997) *Cisco Systems' Tag Switching Architecture Overview*, Rekhter, Y., Davie, B., Katz, D., Rosen, E. and Swallow, G.

RFC 3031 (2001) *Multiprotocol Label Switching Architecture*, Rosen, E., Viswanathan, A. and Callon, R.

RFC 3022 (2001) *Traditional IP Network Address Translator (Traditional NAT)*, Srisuresh, P. and Egevang, K.

RFC 3035 (2001) *MPLS using LDP and ATM VC Switching*, Davie, B., Lawrence, J., McCloghrie, K., Rosen, E., Swallow, G., Rekhter, Y. and Doolan, P.

Stallings, W. (1992) *ISDN and Broadband ISDN*, 2nd edn, Macmillan.

Stallings, W. (1998) *High-Speed Networks: TCP/IP and ATM Design Principles*, Prentice Hall.

Websites

ATM Forum

IETF (Internet Engineering Task Force)

Internet Assigned Numbers Authority

ISO (International Organization for Standardization)

World Wide Web Consortium

Acknowledgements

The content acknowledged below is Proprietary (see [terms and conditions](#)). This content is made available under a

[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 Licence](#)

Course image: © [Kevin Dooley](#) [Creative Commons Attribution 2.0 Licence](#).

<http://www.travelbusy.com/kevin-dooley-photography>.

All materials included in this course are derived from content originated at the Open University.

Don't miss out:

If reading this text has inspired you to learn more, you may be interested in joining the millions of people who discover our free learning resources and qualifications by visiting The Open University - www.open.edu/openlearn/free-courses