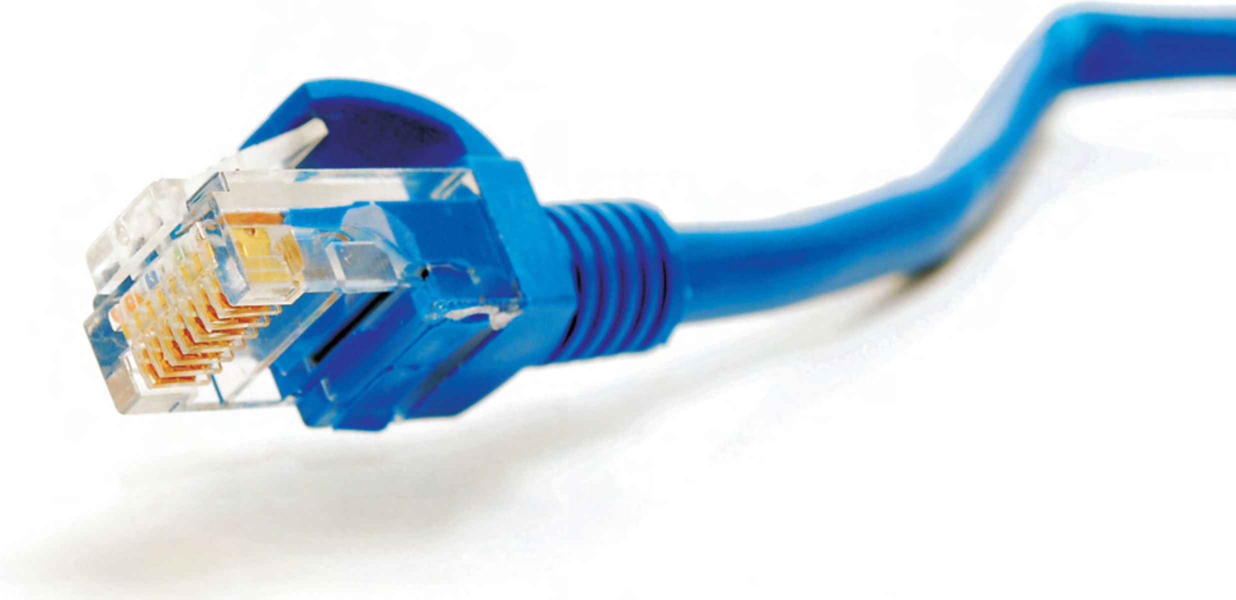


A Complete Guide to Computer Networks

Prof. Rachna Sharma
Prof. Sudipto Das

A Complete Guide to Computer Networks

A Complete Guide to Computer Networks

By

Dr. RACHNA SHARMA

MCA, MTech, (Ph.D)
Associate Professor & Head,
Department of MCA, CMR Institute of Technology, Bangalore, Karnataka

Dr. SUDIPTO DAS

MCA, ADSE(H), (Ph.D)
Assistant Professor,
Department of MCA, CMR Institute of Technology, Bangalore, Karnataka

UNIVERSITY SCIENCE PRESS
(An Imprint of Laxmi Publications Pvt. Ltd.)
BANGALORE • CHENNAI • COCHIN • GUWAHATI • HYDERABAD
JALANDHAR • KOLKATA • LUCKNOW • MUMBAI • RANCHI • NEW DELHI
INDIA • USA • GHANA • KENYA

A COMPLETE GUIDE TO COMPUTER NETWORKS

Printed and bound in India
Typeset at Shubham Composer
First Edition: 2015
ISBN 978-93-81159-68-2

Limits of Liability/Disclaimer of Warranty: The publisher and the author make no representation or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties. The advice, strategies, and activities contained herein may not be suitable for every situation. In performing activities adult supervision must be sought. Likewise, common sense and care are essential to the conduct of any and all activities, whether described in this book or otherwise. Neither the publisher nor the author shall be liable or assumes any responsibility for any injuries or damages arising herefrom. The fact that an organization or Website if referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers must be aware that the Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

All trademarks, logos or any other mark such as Vibgyor, USP, Amanda, Golden Bells, Firewall Media, Mercury, Trinity, Laxmi appearing in this work are trademarks and intellectual property owned by or licensed to Laxmi Publications, its subsidiaries or affiliates. Notwithstanding this disclaimer, all other names and marks mentioned in this work are the trade names, trademarks or service marks of their respective owners.

PUBLISHED IN INDIA BY

UNIVERSITY SCIENCE PRESS
(An Imprint of Laxmi Publications Pvt.Ltd.)
113, GOLDEN HOUSE, DARYAGANJ,
NEW DELHI - 110002, INDIA
Telephone : 91-11-4353 2500, 4353 2501
Fax : 91-11-2325 2572, 4353 2528
www.laxmipublications.com   info@laxmipublications.com

Branches

✆	Bangalore	080-26 75 69 30	
✆	Chennai	044-24 34 47 26,	24 35 95 07
✆	Cochin	0484-237 70 04,	405 13 03
✆	Guwahati	0361-254 36 69,	251 38 81
✆	Hyderabad	040-27 55 53 83,	27 55 53 93
✆	Jalandhar	0181-222 12 72	
✆	Kolkata	033-22 27 43 84	
✆	Lucknow	0522-220 99 16	
✆	Mumbai	022-24 91 54 15,	24 92 78 69
✆	Ranchi	0651-220 44 64	

C—
Printed at:

CONTENTS

SECTION-I: BASIC COMPUTER NETWORKS

SECTION-II: ADVANCED COMPUTER NETWORKS

LIST OF FIGURES

FOREWORD

Over my wide experience in the Teaching field, I have come to believe that a good teacher is one who can provide a balanced mix of theoretical content with its practical aspects. Both the authors, Mrs. Rachna Sharma and Mr. Sudipto Das have excelled in this context and have, in fact, also been an example for their other colleagues.

Their grasp on the subject as well as their natural mode of explanation is very easily understood by the students. This book provides a well-compiled source of knowledge in the field of computer networks, for beginners and advanced users alike. I can easily recommend the book for any learner in this field, as this book can easily be used as a textbook in most universities for BCA, MCA as well as Computers Science branches, as it covers all topics related to Computer Networks, both basic and advanced.

I sincerely hope all their efforts will gather the success that they deserve.

Dr. Y S Kumaraswamy
Professor and Head—Department of MCA
Dayanandasagar College of Engineering
Bangalore, Karnataka

PREFACE

In this fast moving age, Communication is the backbone of life, in all its spheres, and Networks form the essence of any such interchange. Our book, "A Complete Guide to Computer Networks" aims to provide an in-depth understanding on network technologies that have continued to revolutionize the world of communication, ever since.

The book, can be characterized by the following features:

- It provides an in-depth knowledge of the fundamentals of Computer Networks
- It is an excellent guide for even beginners in this field
- It also covers advanced topics, for those who may already be well-versed in the fundamentals
- It uses a very easy language so that any average reader would also be able to benefit from the concepts discussed in the text.

We hope we would continue to provide the readers with relevant knowledge forever and would welcome their appreciation and suggestions in this regard.

—Authors

Section-I

Basic Computer Networks

1

BUILDING APPLICATIONS

Introduction

While trying to understand how to design a network, we need to understand the different aspects related to the application currently being used by us. This also is largely dependent on the kinds of users making use of the network system. In order to effectively manage the design of a network, we would be viewing the concept of network architecture. We would also be discussing the measures, or metrics, used to evaluate the efficiency of computer networks.

1.1 APPLICATIONS

When we visit any web site, say http://www.laxmipublications.com/index, we first specify the URL (Uniform Resource Locator) starting with http (Hyper Text transfer Protocol), which is used to download the page, with www.laxmipublications.com providing the machine name that serves the page and the particular page name (index), identifying the location on the machine to be accessed.

Once we click the URL, almost 17 messages may be exchanged over the Internet, including the page being considered as a single message. Of these, six messages will be used to translate the server name (www.laxmipublications.com) to its Internet address (say 192.35.167.10), next three messages being used to establish a TCP (Transmission Control Protocol) connection within the browser and server, following four messages for your browser to send an HTTP *get* message to the server and the server to reply with the requested page with acknowledgements from either side and the final four messages to terminate the connection.

Another example of Internet application is Streaming Audio & Video. It was possible to have viewed a video file or maybe hear an audio clip over the Internet, but the user had to wait for the entire file to be downloaded before starting to play it. This new technology, known as streaming, allows the user to view the file simultaneously as it is being downloaded. In other terms, it can be called real-time audio and video download.

Videoconferencing is another application becoming increasingly popular over the Internet. We usually would expect that while using the Internet to transfer our Telephone calls, the visual appearance should match the words heard. Any delay between the audio and video transfer rates would render the message inappropriate. Recent technologies have made it possible to enable interactive video applications over the Internet, thus allowing the use of two-way Videoconferencing.

Skype is a popular videoconferencing tool. Figure 1.1 shows the user interface for a Skype session. The others include a Videoconferencing &whiteboard application (PalBee) that allows users tosend sketches and slides to each other, a visual audio tool called vat, and a session directory (sdr) that is used to create and advertise videoconferences. Of these, PalBee works only on Windows platform, Skype on both Windows & Linux platforms and theremaining toolsrun on Unixand Linux platforms.

Even though the examples are limited, there exists a wide variety of applications that can be created over the Internet, either using traditional Web Pages, or through Videoconferencing, thus providing an insight into the complexity of Internet design. Proceeding sequentially from one problem to another, the book explains the evolution of a network that supports this gamut of applications over the World Wide Web.

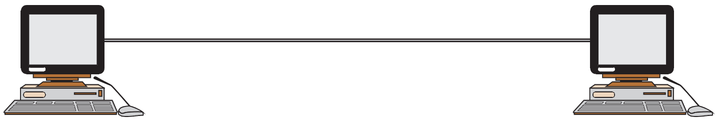
1.2 REQUIREMENTS

The first step is to identify the set of constraints and requirements on which the network design will be designed. Before we start it is important to know clearly what you expect from the network, which however depends on your particular requirements. Some of the common preferences include:

- An application programmer would list the services that his or her applicationneeds, for example, a guarantee that each message the application sends will be delivered without error within a certain amount of time.
- A network designer would list the properties of a cost-effective design, for example, that network resources are efficiently utilized and fairly allocated todifferent users.
- A network provider would list the characteristics of a system that is easy to administer and manage, for example, in which faults can be easily isolated and where it is easy to account for usage.

1.2.1 Connectivity

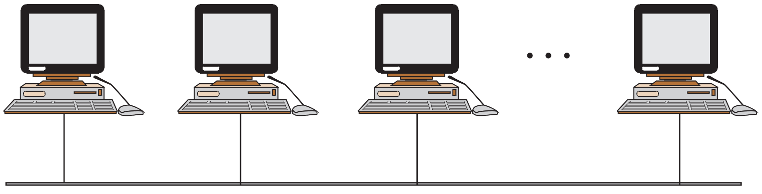
At the lowest level, we need to provide connectivity. It could be using two or more computers directly connected by some physical medium such as a coaxial cable or an optical fiber. In that scenario, computers are known as nodes and the medium which connects those nodes is known as link.



(a)

Figure 1.1(a): Point to point link

It could be point to point or multi-access. Point to point means computers are directly connected to each other. In multi-access link multiple computers can be connected together via one communication link.



(b)

Figure 1.1(b): Multi-access link

Multi-access link has limitation with respect to distance and size. The usage of point to point links is limited, because as the number of nodes increase, the number of wires required to manage the whole system would become quite large and unmanageable. This is more so, because computers can also be connected indirectly, by using an additional link.

The nodes in such a network would be running software that will forward packets from one link to another, in turn forming a *switched* network. The most common switched networks are:

1. Circuit Switched
2. Packet Switched

The Circuit switched method is employed by Telephone System whereas the Packet Switched mechanism is common in cell phone technology.

Packet Switched system uses store and forward technology wherein each node stores the packet in its internal memory and then forwards it to the next node whilea circuit switched network first establishes a dedicated circuit across a sequence of links.A Packet is just a small piece of information that can range from simple text to multimedia information.

In the following figure, the terminal nodes are connected to each other via intermediary nodes, whose task is to forward the packets to the destination node. These are also known as switches. Thus, a network (either point-to-point or multi-access) is represented by a cloud of computers, connected by switches that form the backbone, which would transfer the messages from one node to another, across the cloud (network).

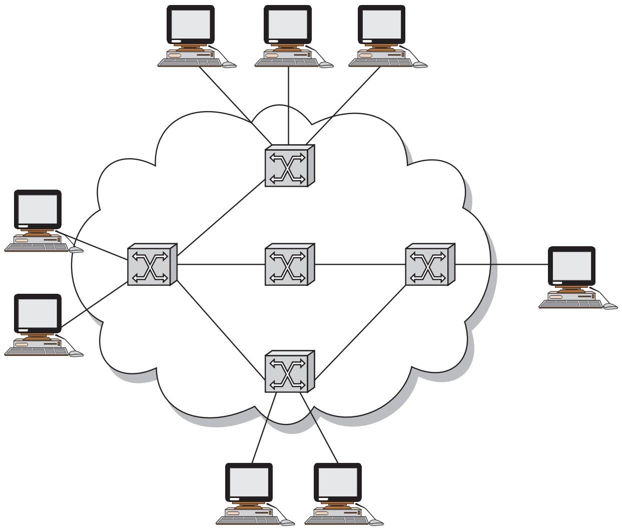


Figure 1.2(a): Switched network

Another way of connecting computers will be with the set of independent networks (clouds) are interconnected to form an internetwork, or internet for short.

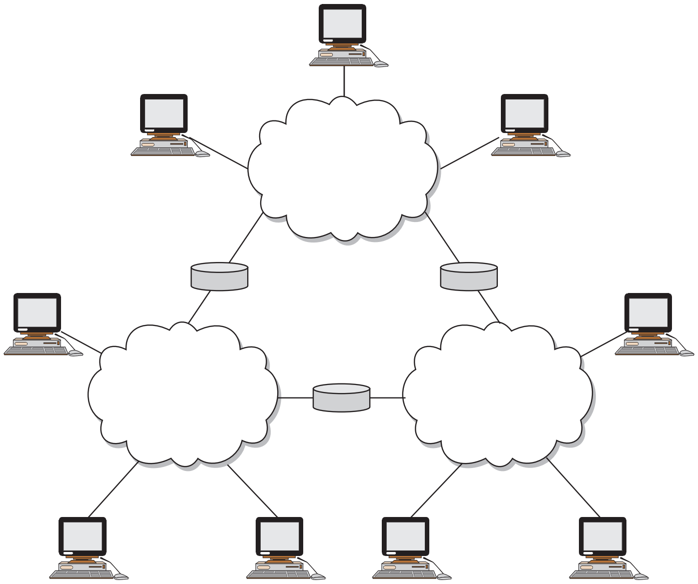


Figure 1.2(b): Internetwork

A node that is connected to two or more networks is known as router or gateway. Large networks can be built using above mentioned heterogeneous networks.

Once the network is built, the next task is to identify the node with which the other node wants to communicate. This could be done by assigning address to each node. When a source node wants to send a message to any destination node, it will first specify the address of the destination node. If the source and destinations are not connected directly then switches and routers can be used to redirect the message to the destination node, making use of the specified address. This process of finding the path to forward the packets based on the destination address is known as routing.

Depending on the range of recipients for the message, transmission could be of three types: Unicast, where a source node sends packets to a single destination, multicast when it sends it to more than one but limited destination nodes and broadcast if it is meant for all nodes in the network.

1.2.2 Cost Effective Resource Sharing

In the above section, we saw that switching could be used in order to propagate messages through the network, even though two needs may not be physically connected. However, in this arrangement, only point-to-point message transfer is possible at a time. To scale this capability to multiple nodes being able to send as well as receive simultaneously, we can use the concept of multiplexing, in addition to switching.

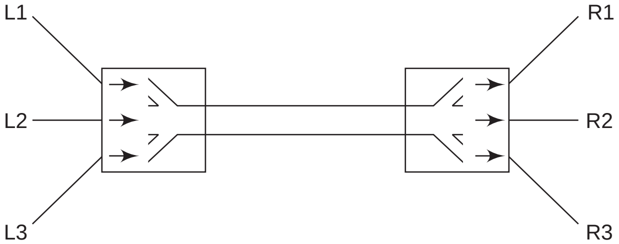


Figure 1.3: Multiplexing to share resources

Consider the case where hosts L1, L2 and L3 want to send messages to R1, R2 and R3 with the help of a shared network.In the above diagram, three flows of data are multiplexed on a single flow by switch1 and can be de-multiplexed into three different flows by switch2.There are various methods available to multiplex several flows into one flow and de-multiplex a single flow into multiple flows. Some of the common methods are described below.

1. **Time Division Multiplexing(TDM)** is a type of digital or (rarely) analogmultiplexing in which two or more signals or bit streams are transferred almost simultaneously, by dividing a single channel into multiple sub-channels, with every sub-channel being accessed separately. This is done by using a time-limiting factor. The time domain is divided into several recurrent timeslots of fixed length, one for each sub-channel. A sample byte or data block of sub-channel 1 is transmitted during timeslot 1, sub-channel 2 during timeslot 2, etc.

   One TDM frame consists of one timeslot per sub-channel plus a synchronization channel and sometimes error correction channel before the synchronization. After the last sub-channel, error correction, and synchronization, the cycle starts all over again with a new frame, starting with the second sample, byte or data block from sub-channel 1, etc.

   Examples:

   The plesiochronous digital hierarchy (PDH) system, also known as the PCM system, for digital transmission of several telephone calls over the same four-wire copper cable (T-carrier or E-carrier) or fiber cable in the circuit switched digital telephone network.

   Limitation: If host has no data to send then its time slot for that particular duration will be wasted. Another limitation is that it is not possible to change quantum of slots, as the number of time slots are allotted beforehand.

2. **Frequency-division multiplexing (FDM)** is a form of signal multiplexing which involves assigning non-overlapping frequency ranges to different signals or to each “user” of a medium.FDMA is the traditional way of separating radio signals from different transmitters.

   Telephone companies usually implement channel banks that can carry thousands of voice circuits multiplexed into them. Such systems make use of L-Carrier and co-axial cables to support long distance transmission.

   Limitation: If the host has no data to send then its frequency channel will be wasted.Another limitation is that it is not possible to add new frequencies, as the number of flows is defined in prior.

3. **STDM, or statistical time division multiplexing, is one method for transmitting several types of** data simultaneously across a single transmission cable or line (such as a T1 or T3 line). STDM is often used for managing data being transmitted via a local area network (LAN) or a wide area network (WAN). In these situations, the data is often simultaneously transmitted from any number of input devices attached to the network, including computers, printers, or fax machines.STDM can also be used in telephone switchboard settings to manage the simultaneous calls going to or coming from multiple, internal telephone lines.

In STDM, as in TDM, the physical linkis shared over time. However, the wastage of time is minimized as transmission is demand-based rather than time-decided. Though, slots have been divided based on time, the system waits for that node only if it has some data to send, otherwise the next node gets an opportunity, and so on. Therefore, if only one flow has data to send, it gets to transmit the data without waiting for its quantum to occurand thus the remaining quanta assigned to the other flows are not wasted.

Statistical multiplexing defines an upper bound on the size of the block of data that each flow is permitted to transmit at a given time. This is done to ensure that all the hosts get their turn to transmit a packet.This limited-size block of data is typically referred to as a packet, to distinguishit from the arbitrarily large message that an application program might want to transmit.

Since the packet size is limited, a host may not be able to send the complete message in one packet.In that case the message is broken into various packets and then it would be transmitting packets back to back. If more than one flow has packets to send, the packets are interleaved on the link and the switch has to decide which packet should be forwarded next.

The switch could be designed to offer servicesin any order such as FIFO (first-in-first-out) or it could be using round robin method. In the above mentioned figure 1, the switch has to multiplex three incomingpacket streams onto one outgoing link.

It is possible the switch will receive packets faster than the shared link can accommodate. In this case, the switch is forced to buffer these packets in its memory. If a switch receives packets faster than it can send them for an extended period of time, then the switch will eventually run out of buffer space, and some packets will have to be dropped. When a switch is operating in this state, it is said to be congested.

1.2.3 Support for Common Services

Networks provide the means for the applications to interact with each other. Since many applications require similar kind of service, it would be better to first create those services and then incorporate those services in those applications. The network can provide a channel, which provides a common service for applications. A Channel is like a pipe connecting one application to another.Channels can be designed based on the service which the application requires.

We need to identify the service which most of the applications would require. The most common type of service is the request for a file, and the process which would return the response for that file.Thus, a **request/reply** channel can be designed for such a service.

The request/reply channel would be used by the file transfer and digital library applications. It would guarantee that every message sent by one side is received by the other side and that only one copy of each message is delivered. The request/reply channel might also protect the privacy and integrity of the data that flows over it, so that unauthorized parties cannot read or modify the data being exchanged between the client and server processes.

Another kind of service which most of the applications would require is message stream channel that could be used by both the video-on-demand and videoconferencing applications, provided it is parameterized to support both one-way and two-way traffic and to support different delay properties.

The message stream channel might not need to guarantee that all messages are delivered, since a video application can operate adequately even if some frames are not received. It would, however, need to ensure that those messages that are delivered arrive in the same order in which they were sent, to avoid displaying frames out of sequence. Thus, network designers will probably be inventing new types of channels—and adding options to existing channels—for as long as application programmers are inventing new applications.

1.3 RELIABILITY

As it is essential that the message be transmitted correctly across the network, it needs to appear reliable by trying to firstly avoid and if not so, to at least detect and if possible rectify the errors that may occur. The first kind of error is that of **corrupted bits**. That is, owing to some physical factor, say lightning or electromagnetic interference etc., several consecutive bits may get disordered. However, such errors are quite rare and can be easily solved by simple bit-reversal mechanisms.

The second class of failure occur at the packet level, where an entire packet is lost by the network. This could be because the packet is discarded as the switch that was supposed to forward it is already overloaded and so it is forced to drop the incoming packet. Another possible reason could be that it was containing an uncorrectable error bit and so has to be dumped.

The third category for failure is at the node level, which may be owing to either a physical link being cut, or the connection fails owing to one of the computers crashing. This could happen due to either a hardware failure, power failure or even a crashed software. Such errors are quite difficult to deal with, owing to the complexity in identifying the real cause of the error and thus can have quite a devastating effect if the problem persists for extended periods of time.

Defining useful channels involves both understanding the applications' requirements and recognizing the limitations of the underlying technology. The challenge is to fill in the gap between what the application expects and what the underlying technology can provide. This is sometimes called the semantic gap.

Computer network must provide general, cost-effective, fair, and robust connectivity among a large number of computers. network designers have developed general blueprints—usually called a network architecture—that guide the design and implementation of networks.When the system gets complex, the system designer introduces another level of abstraction. The idea of an abstraction is to define a unifying model that can capture some important aspect of the system, encapsulate this model in an object that provides an interface that can be manipulated by other components of the system, and hide the details of how the object is implemented from the users of the object. We were providing an abstraction for applications that hides the complexity of the network from application writers.

The term architecture is commonly used to explain network. Network architecture explains what things exist, how they operate, and what form they take. Architecture includes hardware, software; data link controls (DLCs), standards, topologies, and protocols. The concept of protocol architecture is that there must be a high degree of cooperation between the two computers. This task is broken into subtasks, instead of single module. As given in the figure below three modules are used. There is structured set of modules that implements the communication function.

That structure is referred to as protocol architecture.

The following figure illustrates a simplified architecture for file transfer.

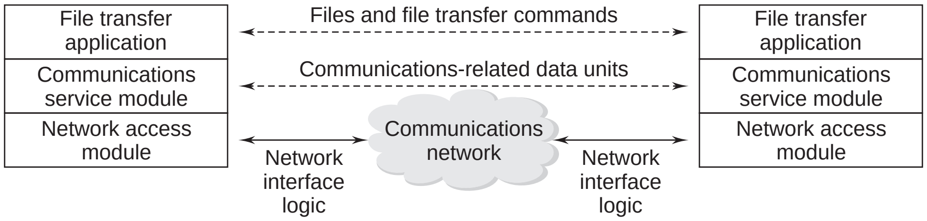

File transfer application
Files and file transfer commands
File transfer application
Communications service module
Communications-related data units
Communications service module
Network access module
Communications network
Network access module
Network interface logic
Network interface logic

Figure 1.4: Network architecture for file transfer

In information technology, protocol is an agreement between the communicating parties on how communication is to proceed. For example; if a women is introduced to a man, he may decide to shake her hand or can simply say 'hello' depending on which country they belong to. Violating the protocol will make communication more difficult, if not completely impossible. A protocol is the special set of rules that end points in a telecommunication connection. There are protocols for the data interchange at the hardware device level and protocols for data interchange at the application program level. In the standard model known as Open Systems Interconnection (OSI), there are one or more protocols at each layer in the telecommunication exchange that both ends of the exchange should identify and accept. As agreed upon format for transmitting data between two devices, the protocol decides the following,

- The types of error checking to be used
- Data compression method, if any
- How the sending device will indicate that it has finished sending a message
- How the receiving device will indicate that it has received a message

1.3.1 Protocol Hierarchies

Modern networks are using the concept of layered protocol. The early communications were simple and did not use layers. Networks are arranged as a stack of **layers** or **levels** to reduce the complex design.

Each level is built upon the one below it. Each network varies in number, name, contents, and function of each layer. The function of each layer is to offer some particular services to the higher layers such as protect the higher layers from the details of how the offered services are actually implemented.

In short, each layer is offering certain services to the layer above it. This basic concept is used in computer science. In computer this concept is known as information hiding, abstract data types, data encapsulation, and object-oriented programming. A software or hardware offers service to its users but keeps the aspects of its internal position and algorithms hidden from them.

Layer *n* on one computer carries information to layer *n* on another computer. The rules and conventions used in this communication are known as the layer *n* protocol. A protocol is an agreement between the communicating parties on how communication is to proceed.

If we go against the protocol, it will make communication more difficult or make it impossible to communicate. In the figure given below, a five-layer network is explained. The entities consist the related layers on different systems are called *peers*. The peers can be processes, hardware devices, human being or anybody or anything communicate by using the protocol.

Practically data are not directly transmitted from layer *n* to layer *n* on another machine. Each layer passes data to the layer immediately below it and controls the data until it reaches the lowest layer. Physical medium is available below layer 1 and communication occurs through this medium. In the figure given above, virtual communication is shown by dotted lines and physical communication is shown by solid lines.

An interface is available between each pair of neighboring layers. The interface describes what operations and services the lower layer offers to upper one.

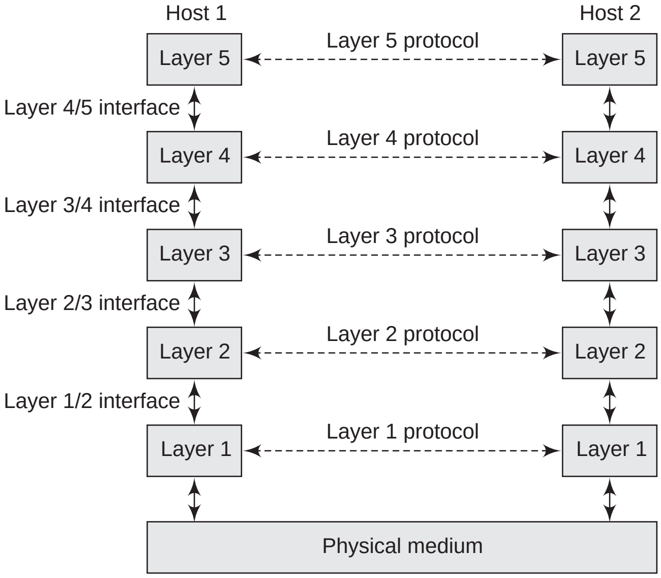


Figure 1.5: Layered network architecture

It is important to give sufficient consideration to define interfaces between the layers because each network designers offer different number of layers and different tasks for each layer. If the interface is perfect, the functions of the layers are also will be trouble free.

A set of layers and protocols is called network architecture. The design of the architecture should have sufficient information to allow an implementer to write the program or build the hardware for each layer. The details of the implementation and the specification of the interfaces are hidden in the system and not visible from the outside and these details are not part of the architecture. It is not necessary that the interfaces on all machines in a network be the same, provided that all machines can correctly use all the protocols. A list of protocols used by a certain system is called a protocol stack.

Example: The figure given below explains how to provide communication to the top layer of the five-layer network. The data S is produced by an application process running in layer 5 and pass it to layer 4 for transmission. Layer 4 puts a header in front of the message to identify the message and passes the result to layer 3. The header consists of control information, such as sequence numbers to allow layer 4 on the destination system to deliver data in the right sequence. Some headers has the details of sizes, times, and other control fields. For layer 4 protocols, most of the networks have no limit to the size of data transmitted but there is a limit required by the layer 3 protocol. Hence, in layer 3, the data is split into smaller units.

According to this example, S is split into two parts such as S1 and S2. Layer 3 decides which of the outgoing lines to use to pass the packets to layer 2. Layer 2 adds not only a header to each piece, but also a trailer, and gives the resulting unit to layer 1 for physical transmission. At the receiving machine the data moves upward with headers being deprived as it progresses. None of the headers for layers below *n* are passed up to layer *n*.

From the figure above we can understand the connection between the virtual and actual communication and the difference between protocols and interfaces. For example; the peer processes in layer 4 naturally think to communicate in horizontal way by using the layer 4 protocols. But the transmission happens with lower layers through the ¾ interface.

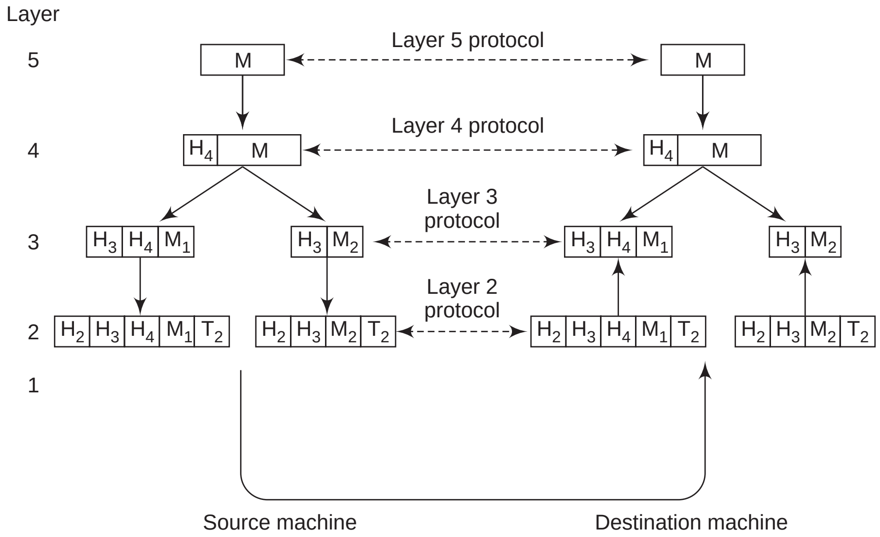


Figure 1.6: Data transfer through layers

1.3.2 The OSI Reference Model

In this section we will talk about two important network architectures; the OSI reference model and the TCP/ IP reference model. Open System Interconnection (OSI) is a standard reference model for communication between two end users in a network. This model is based on an application developed by the International Standard Organization (ISO). Since this model is deals with open systems, that is, systems that are open for communication with other systems, we call it as OSI model in short. OSI model is divided into seven layers. We can divide the seven layers into two groups. The upper four layers are used whenever a message passes from or to a user. The lower three layers are used when any message passes through the host computer.

The standard that are applied to these layers are that the layer boundaries should be chosen to minimize the information flow across the interfaces and it should have enough number of layers so that lot of functions should not throw together in the same layer but at the same time it should be small enough that the architecture does not become large.

The other standard applied to these layers is that all the layers must perform a precise function and while selecting the function of each layer, we should consider internationally standardized protocols. A layer should be created where a different concept is needed.

OSI divides into seven layers. OSI model itself is not network architecture because it does not specify the exact services and protocols to be used in each layer. It informs what each layer should do. ISO has produced standards for all the layers, but these are not part of the reference model itself. Each one has been published as a separate international standard. In the following section we will discuss each layer of the model.

OSI Reference Model Architecture

The Physical Layer—This layer transmits the bit stream through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier. This layer is responsible to transmit raw bits over a communication channel. The design should be perfect when one side sends a 1 bit; it is received by the other side as a 1 bit, not as a 0 bit.

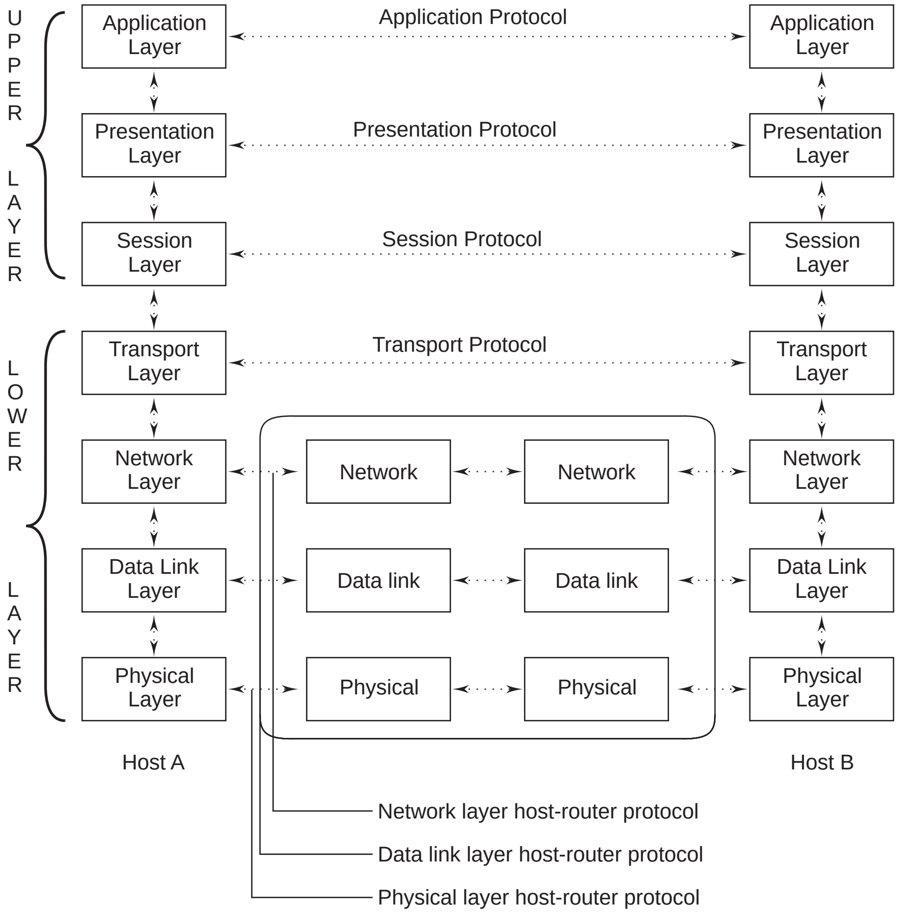


Figure 1.7: Layer-wise description of protocols

The Data Link Layer – The principle service provided by the data link layer to higher layers is that of error detection and control. It achieves this function by having the sender break up the input data into data frames and transmits the frames sequentially. Then the receiver confirms correct receipt of each frame by sending back an acknowledgement frame if the service is consistent. The data link layer is divided into two sub-layers; The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sub-layer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking.

The Network Layer – This layer handles the routing of the data that is sending it in the right direction to the right destination on outgoing transmissions and receiving incoming transmissions at the packet level. This layer does routing and forwarding. The network layer controls the operation of the subnet. If many packets are presented in the subnet simultaneously they will create problem. Network layer takes control of such jamming also. In general a network layer is responsible for the quality of service provided. In general a network layer is responsible for the quality of service provided.This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.

The Transport Layer – This layer controls the end-to-end issue, such as determining whether all packets have arrived and error checking. It makes sure entire data transfer. The main task of this layer is to accept data from above layer, split it up into smaller units if needed and then pass these to the network layer. The layer ensures that the pieces all arrive correctly at the other end. The transport layer decides what type of service to provide to the session layer, and ultimately to the users of the network and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.

The Session Layer – This layer deals with session and connection coordination. It sets up, coordinates, and terminates conversations, exchanges, and dialogs between the applications at each end. This layer allows users on different systems to establish sessions between them. This layer offers other services like dialog control that is keeping track of whose turn it is to transmit and token management that is preventing two parties from attempting the same critical operation at the same time and synchronization that is ensuring long transmission to permit them to continue from where they were after a crash.

The Presentation Layer — This layer is part of an operating system, which converts incoming and outgoing data from one presentation format to another; for example from a text stream into a popup window with the newly arrived text. It is called as syntax layer too. The lower layers are mostly concerned with moving bits around but the presentation layer is concerned with the syntax and semantics of the information transmitted. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer.

The Application Layer – This layer has a variety of protocol that is commonly needed by users. HTTP (Hyper Text Transfer Protocol) is the basis for the World Wide Web and this protocol is widely used application protocol. This is the layer at which communication partners are identified, quality of service is identified, and user authentication and privacy are considered. This layer supports application and end-user processes. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services. Telnet and FTP are applications that exist entirely in the application level.

1.3.3 The TCP/ IP Reference Model

The TCP/IP reference model is the network model used in the present Internet architecture .It has its origins back in the 1960's with the grandfather of the Internet, the ARPANET. This was a research network sponsored by the Department of Defense in the United States. The major goals are:

- Ability to connect multiple networks together.
- Ability for connections to remain perfect as long as the source and destination machines were functioning.
- To be built on flexible architecture.

This reference model got name from its main two protocols, TCP (Transmission Control Protocol) and IP (Internet Protocol). They choose to build a packet-switched network based on a connectionless internet work layer.

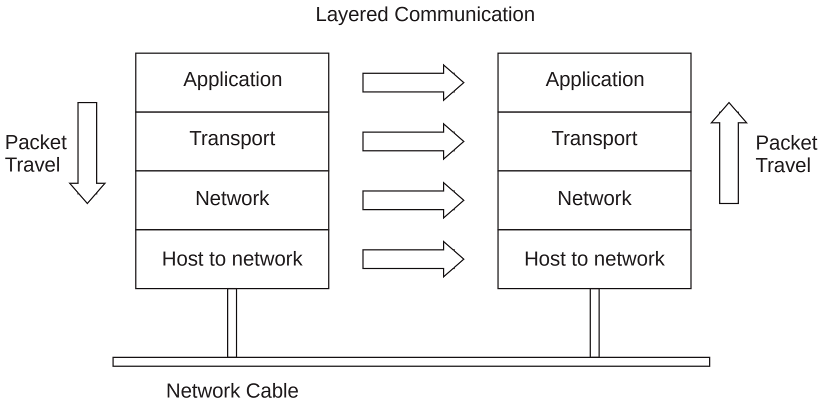


Figure 1.8: Layered communication

TCP/IP Network Protocol

The basic idea of the networking system is to permit one application on a host computer to communicate to another application on a different host computer. The application creates its request and passes down to the lower layers that add their own control information, either a header or footer on to the packet. At last the packet reaches the physical layer and is transmitted through the cable onto the destination host. The packet travels through the different layers with each layer reading and removing the header or footer, which was attached by its parallel system. Finally the packet reaches the application it was destined for.

1.4 PERFORMANCE

Bandwidth and Latency

Network performance is measured in two fundamental ways: bandwidth (also called throughput) and latency (also called delay).

- **Bandwidth** — Bandwidth is the frequency range of a channel, measured as the difference between the highest and lowest frequencies that the channel supports. The maximum transmission speed is dependent upon the available bandwidth. The larger the bandwidth, the higher the transmission speed. For example, a network might have a bandwidth of 10 million bits/second (Mbps), meaning that it is able to deliver 10 million bit per second.

  We can think of a second of time as some distance that could be measured with a ruler and the bandwidth as how many bits could fit in that distance.
- **Latency** — The second performance metric, latency, corresponds to how long it takes a message to travel from one end of a network to the other. Latency is measuredstrictly in terms of time. The term latency refers to any of several kinds of delays typically incurred in processing of network data. A so-called low latency network connection is one that generally experiences small delay times, while a high latency connection generally suffers from long delays.

When loading a Web page, for example, most satellite users can observe a noticeable delay from the time they enter a Web address to the time the page begins loading. This high latency is due primarily to **propagation delay** as the request message travels at the speed of light to the distant satellite station and back to the home network. Once the messages arrive on Earth, however, the page loads quickly like on other high-bandwidth Internet connections (DSL or cable).

Besides propagation delays, latency also may also involve transmission delays (properties of the physical medium) and processing delays (such as passing through proxy servers or making network hops on the Internet).

We could define the total latency as:

Latency = Propagation + Transmit + Queue
Propagation = Distance/Speed of Light
Transmit = Size/Bandwidth

where Distance is the length of the wire over which the data will travel, Speed of Light is the effective speed of light over that wire, Size is the size of the packet, and Bandwidth is the bandwidth at which the packet is transmitted.

Bandwidth and latency combine to define the performance characteristics of a given link or channel.

Example –Numerical

Consider point-to-point link 20km in length. At what bandwidth would propagation delay (at a speed of $2*10^8$ m/sec) equal transmit delay for 100-byte packet? What about 512-byte packet?

Given Data,

Distance = 20km=20,000m
Speed = $2*10^8$ m/sec

Given, Propagation delay = Transmit Delay

Distance/Speed=Size/Bandwidth

For a Size of 100 byte

$20,000/2*10^8$ m/sec=100 bytes/Bandwidth

→ Bandwidth = 800*2*10=8 Mbps
20,000

For a size of 512 Bytes

$20,000/2*10^8$ m/sec = 512*8 bits/Bandwidth
Bandwidth = $2*10^8*512*8$
20,000
= 40.96 Mbps.

SUMMARY

- Computer networks like the Internet have experienced enormous growth over the past decade and are now positioned to provide a widerange of services—remote file access, digital libraries, video conferencing—to hundreds of millions of users.
- The first step we have taken toward this goal is to carefully identify exactly what we expect from a network. For example, a network must first provide cost effective connectivity among a set of computers.
- This is accomplished through a nested interconnection of nodes and links, and by sharing this hardware base through the use of statistical multiplexing. This results in a packet-switched network, on top of which we then define a collection of process-to-process communication services.

- The second step is to define a layered architecture that will serve as a blueprint for our design.
- An architecture includes hardware, software, data link controls (DLCs), standards, topologies, and protocols.
- OSI model is deals with open systems, that is, systems that are open for communication with other systems.
- OSI model is divided into seven layers.
- The TCP/IP reference model is the network model used in the present Internet architecture
- The TCP/IP model does not clearly differentiate the concept of service, interface, and protocol.
- The TCP/IP model is practically nonexistent but the protocols are widely used.

2

Direct Link Network

Introduction

The most primary type of network is one where all the hosts are directly connected by a physical medium; say, wire or fiber, covering a small area, such as an office building or large enough to be spreading across entire continents. In either case, the first step would however be to first connect the participating nodes.

In addition, there are five issues that must be taken care of before the nodes can successfully exchange packets.

First, the bits need to be converted into a form that the medium (wire or fiber) can carry, ensuring that it is meaningful to the receiver. This is also known as ***encoding***.

Next is to specify the combination of individual bits into complete messages that can be delivered to the recipient. These messages are often known as frames and the process is called ***framing***.

The third issue that arises is owing to the fact that there could always be an error if the frames get corrupted during transmission. Thus we need an ***error-detection*** module to take care of that. Even though there may have been a frame getting corrupted occasionally, that should not disrupt the delivery of the packet concerned.

In fact, delivery can be delayed, but it should be completed. Thus, the fourth issue that arises is that of ***reliable delivery***.

The penultimate case occurs when in contrast to a basic point-to-point link, a link needs to be shared amongst multiple hosts and there arises the need to arbitrate access to this link. This is commonly referred to as the ***media access control*** problem and is a major factor in any network design.

Though these five issues can be discussed as just simple topics, they are in fact quite deep problems and are addressed in different ways by different networking technologies.

This chapter considers these issues in the context of four specific network technologies:

Point-to point links, Carrier Sense Multiple Access (CSMA) networks (of which Ethernet is the most famous example), Token ring (of which IEEE Standard 802.5 and FDDI are the most famous examples) and Wireless (for which 802.11 is examining the building blocks that will be used; i.e., nodes and links).

We then explore the first three issues—**encoding, framing,** and **error detection**—in the context of a simple point-to-point link. The techniques introduced here are quite general and therefore apply equally well to multiple-access networks. Next, we discuss the problem of **reliable delivery**, only for point-to-point links, since link-level reliability is usually not implemented in shared-access networks. Finally, we address the **media access** problem in the context of CSMA, Token rings, and Wireless.

These five functions are usually implemented in a Network Adaptor card that provides the interface between the computer hardware and the physical media. Further, an adaptor is controlled by device driver software running on the nodes, or workstations. Therefore, *bits are exchanged between adaptors while correct frames are exchanged between nodes*.

2.1 COMMUNICATION HARDWARE

For transmitting and receiving we need two devices and these devices need hardware to communicate with each other. In this section we would discuss about the various hardware such as;

- Sender and Receiver Hardware
- Communication Devices
- Communication Channels

2.1.1 Nodes

Data communication is accomplished by using several communication devices. Software is used for data exchange. The devices used to communicate across a data communication network are called workstations.

Nodes are usually general-purpose computers, like a desktop workstation, a multiprocessor or a PC. Let us assume it to be a workstation-class machine. This workstation could serve as a ***host*** that helps users to run their application programs, or as a ***switch*** that forwards messages from one link to another or it may be even configured to act as a ***router*** which would forward internet packets across networks.

Additionally, in case of non-host workstations, we may implement it using special-purpose hardware, usually due to reasons of performance and cost. It is generally possible to build custom hardware that performs a particular function faster and cheaper rather than a general purpose computer that could perform a similar task.

In such a case, we will first describe the basic function being performed by the node as though it is being implemented in software on a general-purpose workstation, and then explain why and how this functionality might instead be implemented by special hardware. It is also helpful to have this knowledge so that we can correctly judge how well the network performs. The following figure depicts a simple block diagram of the workstation-class machine.

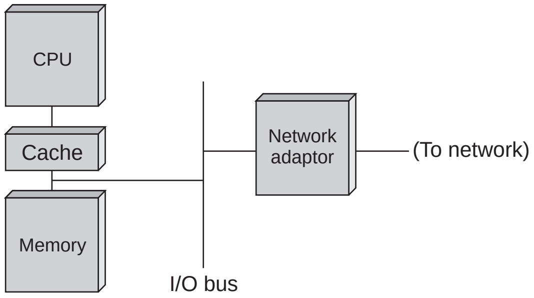


Figure 2.1: Block diagram of a workstation

Each node connects to the network via a network adaptor. This adaptor generally sits on the system's I/O bus and delivers data between the workstation's memory and the network link.

A software module running on the workstation—thedevice driver—manages this adaptor.

2.1.2 Links

Network links are implemented on a variety of different physical media, including **twisted pair** (the wire that your phone connects to), **coaxial cable** (the wire that yourTV connects to), **optical fiber** (the medium most commonly used for high-bandwidth, long-distance links), and **space** (through which radio waves, microwaves, and infrared beams propagate). Physical media is used to propagate signals.

These signals are actually electromagnetic waves traveling at the speed of light. (The speed of light is, however, medium-dependent; electromagnetic waves traveling through copper and fiber do so at about two-thirds the speed of light in vacuum.)

Properties of Electromagnetic Signals

- **Frequency —** It is the rate of change the signal undergoes every second, expressed in Hertz (Hz), or cycles per second. A 30 Hz signal changes thirty times a second. In speech, we refer to it as the number of vibrations per second, as the air that we expel out of our mouth pushes the surrounding air at a specific rate.

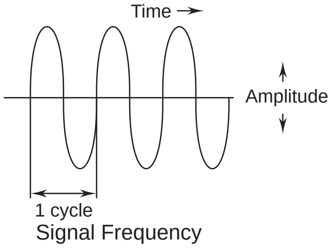


Figure 2.2: Signal frequency

- **Wavelength:** The distance between a pair of adjacent maxima and minima of a wave, typically measured in meters, is called its *wavelength*. Since all electromagnetic waves travel at the speed of light, that speed divided by the wave's frequency is equal to its wavelength.

Links provide the foundation for transmitting all sorts of information, including the kind of data we are interested in transmitting; *i.e.*, binary (1s and 0s). **Data encoding** is the method by which certain communication devices encode digital data onto an analog signal for transmission. For example, telephone lines are analog but computer generates digital data; thus, computers need to convert the digital data into analog data before it can be transmitted through telephone lines.

A digital signal is a series of separate and discontinuous voltage pulses. Each pulse represents a signal element. Binary data are transmitted by encoding data bit into signal elements. There is one-to-one association between bits and signal elements. Generally binary 0 is represented by a lower voltage level and binary 1 by a higher voltage level.

We can think of encoding as being divided into two layers. The lower layer is concerned with ***modulation***—varying the *frequency*, *amplitude*, or *phase* of the signal to effect the transmission of information. **Modulation rate** is the rate at which the signal level is changed and it is expressed in ***baud***.

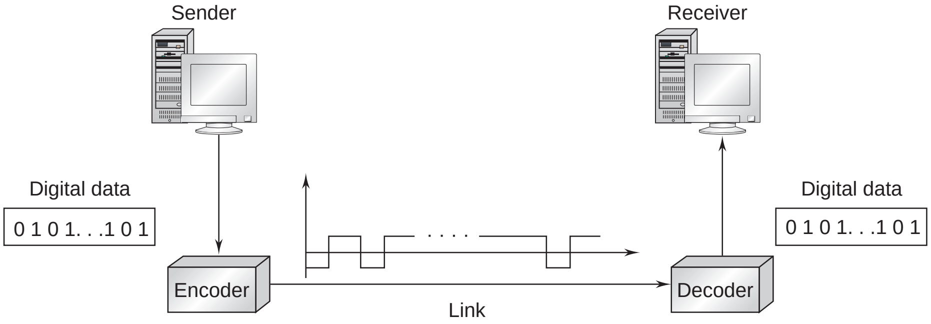


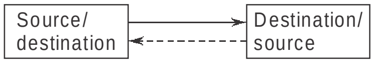
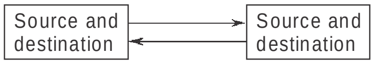
Figure 2.3: Encoding and decoding for data transfer

Another attribute of a link is how many bit streams can be encoded on it at a given time. The transmission of information will be either unidirectional or bidirectional. The unidirectional systems that transmit in one direction are called *simplex.* The bidirectional systems that are able to transmit in both directions are called *duplex* systems.

A duplex communication system is comprised of two connected devices that can communicate with one another in both directions. Duplex systems are used in almost all communications networks.

Table 2.1: Communication system attributes

Terminology	Description		Example
Simplex	In this method the signal is transmitted in just one direction Simplex channels are not often used because it is not possible to send back error or control signals to the transmit end.	Source Destination	1. An example of a simplex channel in a computer system is the interface between the keyboard and the computer. Here, key codes need only be sent one way from the keyboard to the computer system. 2. TV signals 3. Paging system that allows a user to receive only alpha-numerical messages.

Half duplex	In half-duplex system, communication is possible on both directions, but only one direction at a time. If one system is receiving a signal, it must wait for the transmitter to stop transmitting, before replying.		1. It is like a one-lane bridge where two-way traffic must give way in order to cross. Only one end transmits at a time, the other end receives. 2. Walkie-talkie
Full duplex	In full-duplex system, communication is possible on both directions. Unlike half-duplex, full-duplex allows communication on both sides simultaneously. Telephone networks are full-duplex since they allow both callers to speak and be heard at the same time.		1. It is like a two lane bridge on a two-lane highway. 2.Telephone 3. Mobile phone

2.1.2.1 Cables

If the nodes we want to connect are in the same room, in the same building in a same campus, then you can buy a piece of cable and physically string it between the nodes. The type of cable you choose to install depends on the technology you plan to use to transmit data over the link. Of these, Category 5 (Cat-5) Twisted pair, that uses a thicker gauge than the twisted pair you find in your homes, is usually considered the indoor networking norm. Because of the difficulty and cost in pulling new cables through a building, every effort is made to make new technologies use the existing cable. Gigabit Ethernet, for example, has been designed to run over Cat-5 wiring. Fiber is typically used to connect buildings at a corporate site.

Table 2.2: Cable media characteristics

Cable	Typical Bandwidths	Distances
Category 5 twisted pair	10 – 100 Mbps	100 m
Thin-net coax	10 – 100 Mbps	200 m
Thick-net coax	10 – 100 Mbps	500 m
Multimode fiber	100 Mbps	2 km
Single-mode fiber	100 – 2400 Mbps	40 km

2.1.2.2 Leased Line

If the two nodes you want to connect are on opposite sides of the country, or even across towns, then it is not practical to install the link yourself. Your only option is to lease a dedicated link from the telephone company, which is quite common nowadays with several providers being available. These are dedicated analog lines or digital lines. Dedicated digital lines are called digital data service (DDS) lines. A modem is used to connect to analog lines, and a Channel Service Unit/Data Service Unit or Digital Service Unit (CSU/DSU) is used to connect to digital lines. The DSU connects to the LAN and the CSU connects to the line.

T Carrier lines — Multiplexers are used to allow several channels on one line. The T1 line is basic T Carrier service. The available channels may be used separately for data or voice transmissions or they may be combined for more transmission bandwidth. The 64Kbps data transmission rate is referred to as DS-0 (Digital Signal level 0) and a full T1 line is referred to as DS-1.

Table 2.3: Characteristics of communication lines

Signal	System	Total Kbps	Channels	Number of equivalent T1 lines
DS-1	T1	1544	24	1
DS-2	T2	6312	96	4
DS-3	T3	44736	672	28
DS-4	T4	274760	4032	3668

T1 and T3 lines are the most common lines in use today. T1 and T2 lines can use standard copper wire. T3 and T4 lines require fiber-optic cable or other high-speed media. These lines may be leased partially called fractional T1 or fractional T3 which means a customer can lease a certain number of channels on the line. A CSU/DSU and a bridge or router is required to connect to a T1 line.

2.1.2.3 Last-Mile Link

The last mile that is the end-link between the consumers and the provider network is often stated in terms of the "last-mile problem" because this aspect of connectivity has proved to be disproportionately expensive to solve; typically, connecting a home PC to an existing network. This means they are probably not suitable for use in building a complete network from scratch, but if you've already succeeded in building a network—and "you" happen to be either the telephone company or the cable company—then you can use these links to reach millions of customers.

The first option is a conventional modem over the POTS (plain old telephone service). Today it is possible to buy a modem that transmits data at 56 Kbps over a standard voice-grade line for less than a hundred dollars. The technology is already at its bandwidth limit, however, which has led to the development of the second option, ISDN.

2.1.2.4 ISDN

ISDN which stands for Integrated Services Digital Network, is a system of digital phone connections allows voice and data to be transmitted simultaneously across the world using end-to-end digital connectivity. The key feature of ISDN is that it integrates speech and data on the same lines, adding features that were not available in the classic telephone system. An ISDN connection includes two 64-Kbps channels, one that can be used to transmit data and another that can be used for digitized voice. (A device that encodes analog voice into a digital ISDN link is called a CODEC, for coder/decoder.) When the voice channel is not in use, it can be combined with the data channel to support up to 128 Kbps of data bandwidth.

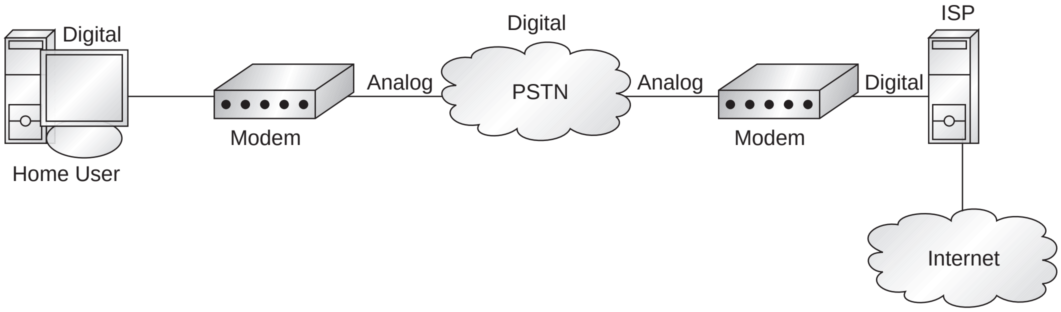


Figure 2.4: ISDN

2.1.2.5 DSL

DSL (Digital Subscriber Line) operates over standard copper telephone lines like dial-up service, but is many times faster than dial-up. In addition to being faster than dial-up, DSL does not tie up the phone line. Coexisting with telephone service in this way allows users to surf the Net and use the phone at the same time.

DSL service requires a DSL modem, which connects to the telephone wall jack and computer. The device acts as a modulator, translating the computer's digital signals into voltage sent across the telephone lines to a central hub known as a Digital Subscriber Line Access Multiplier (DSLAM).

In lay terms the DSLAM acts as a switchboard for local DSL clients, routing requests and responses between each client's computer address and the Internet.It is actually a collection of technologies that are able to transmit data at high speeds over the standard twisted pair lines.

2.1.2.6 ADSL

ADSL (Asymmetric Digital Subscriber Line) is a high-speed Internet access service that utilizes existing copper telephones lines to send and receive data at speeds that far exceed conventional dial-up modems.

The fastest dial-up modems are rated at 57 kilobits per second (Kbps), and usually operate at about 53 Kbps under good conditions. By comparison, ADSL allows data stream speeds from 1.5 to 8 megabits per second (Mbps), depending on the grade of ADSL service purchased.

ADSL uses standard telephone lines to transmit upstream and downstream data on a digital frequency, which sets these data streams apart from the analog signals telephones and fax machines use. Because the ADSL signal is operating on a different frequency, the telephone can be used normally, even when surfing the Web with ADSL service. The only requirement will probably be inexpensive DSL filters on each phone or fax line, to remove any "white noise" on the line that might be generated from ADSL service.

The "asymmetric" in ADSL refers to the fact that the downstream data rate, or the data coming to your computer from the Internet, is traveling faster than upstream data, or the data traveling from your computer to the Internet. Upstream data rates are slower because Web page requests are fairly miniscule data strings that do not require much bandwidth to handle efficiently.

ADSL service requires an Internet service provider (ISP), and ADSL modem. The modem is often provided free of charge, and most ISPs that offer ADSL service require subscriber contracts of one year. ADSL is an "always on" service, meaning that as long as your computer is powered on, it will automatically stay connected to the Internet unless you manually disconnect via software or hardware.

2.1.2.7 VDSL

VDSL (Very high bit-rate Digital Line Subscriber) is next generation DSL at super-accelerated rates of 52 Mbps (megabits per second) downstream and 12 mbps upstream. Downstream data rates refer to download speeds, or the speed at which data travels to your computer, while upstream data rates refer to upload speeds, or the speed at which data travels from your computer to the Internet.The telephone company would have to put VDSL transmission hardware in neighborhoods, with some other technology (*e.g*., STS-*N* running over fiber) connecting the too high an error rate.

As with other broadband technologies, end-user speeds will depend upon the distance of the feed or loop to the local telephone company office. Shorter distances afford faster rates, while longer loops degrade signal and speed. One drawback of VDSL is that it requires a very short loop of about 4000 feet (1219 meters), or three-quarters of a mile. However, another complication can inadvertently create a solution for the distance problem: the complication of fiber optic lines.

2.1.2.8 Cable Modems

A cable modem is a device that enables you to connect your PC to a local cable TV line and receive data at about 1.5 Mbps. This data rate far exceeds that of the prevalent 28.8 and 56 Kbps telephone modems and the up to 128 Kbps of Integrated Services Digital Network (ISDN) and is about the data rate available to subscribers of Digital Subscriber Line (DSL) telephone service.

A cable modem can be added to or integrated with a set-top box that provides your TV set with channels for Internet access. In most cases, cable modems are furnished as part of the cable access service and are not purchased directly and installed by the subscriber. The actual bandwidth for Internet service over a cable TV line is up to 27 Mbps on the download path to the subscriber with about 2.5 Mbps of bandwidth for interactive responses in the other direction.

2.1.2.9 Wireless Link

Wireless communications are rapidly expanding all across the globe, with technology becoming more widespread and cost-effective. A range of standards govern the exchange of messages across wireless links with the initial being the AMPS or Advanced Mobile Phone System (AMPS). AMPS has been the standard for cellular phones in the United States, since the establishment of such systems, and is based on analog modes.

However, with the digital transformation of services, PCS (Personal Communication Services) became more popular and has now emerged into the GSM (Global System for Mobile Communication) for almost the entire world.

All three systems currently use a network of towers to transmit signals, though efforts have been made to support this infrastructure by installing a range of satellites.

Several projects such as ICO, Globalstar, Iridium, and Teledesic have been installed to try and provide these services across the globe, focusing more on reaching those regions that are not currently being serviced by cellular operators.

At the local level, such as inside office buildings, coffeeshops, building complexes, and campuses, wireless links are being used to enhance short-range communications by transmitting signals over the radio and infrared frequencies. In the case of infrared, signals with wavelengths in the 850–950-nano meter (nm) range can be used to transmit data at1-Mbps rates over distances of about 10 m. This technology does not require line of sight, but is limited to in-building environments.

In the case of radio, several different bands are currently being made available for data communication, up to high frequency levels of 17 GHz. Another interesting development in this area is the Bluetooth radio interface operating in the 2.45-GHz frequency band. Bluetooth is designed fordistances up to 10 m with a bandwidth of 1 Mbps. It aims at being used in all devices (*e.g.*, printers, workstations, laptops, projectors, PDAs, mobile phones), thereby eliminating the need for wires and cables in the office. Networks of such devices are beginning to be called piconets.

2.2 ENCODING

Signals propagate over physical links. The task, therefore, is to encode the binary data that the source node wants to send into the signals that the links are able to carry, and then to decode the signal back to the corresponding binary data at the receiving node. The network adaptor contains a signaling component that actually encodes bits into signals at the sending node and decodes signals into bits at the receiving node.There is one-to-one association between bits and signal elements. Generally binary 0 is represented by a lower voltage level and binary 1 by a higher voltage level.

Thus, as illustrated in Figure 2.5, signals travel over a link between two signaling components, and bits flow between network adaptors.

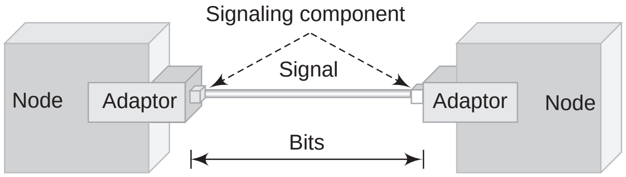


Figure 2.5: Signals travel between signaling components; bits flow between adaptors

Encoding Formats

An encoding format describes the conversion of data bits to signal elements. The common encoding formats are depicted in the following table;

Non-return to Zero-Level (NRZ-*l*) — The easiest way to transmit digital signals is to use two different voltage levels for the two binary digits. For example; the absence of voltage can be used to represent binary 0 and positive voltage can be used to represent binary 1. NRZ-*l* is used to produce digital data by terminals and other devices. If a different code is to be used for transmission, it is generated from an NRZ-*l* signal.

Non-return to Zero Inverted — This encoding technique is an example of *differential* encoding. The information to be transmitted is represented in terms of the changes between successive data symbols in differential encoding. For example; if it is a binary 0 then the bit is encoded with the same signal as the previous bit; if the bit is a binary 1, then the bit is encoded with a different signal than the previous one.

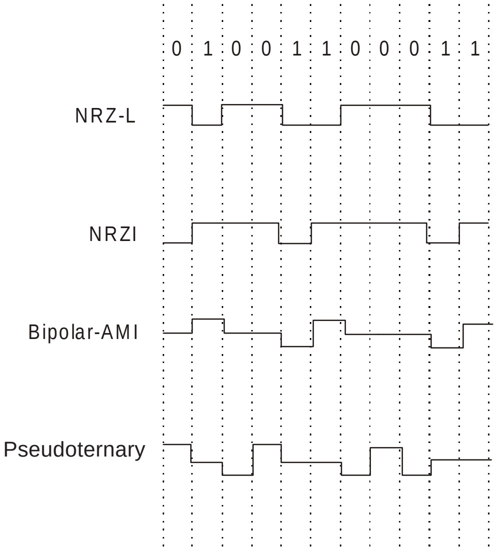
Non-return to Zero-Level (NRZ-L) 0 = high level 1 = low level	 0 1 0 0 1 1 0 0 0 1 1 NRZ-L NRZI Bipolar-AMI Pseudoternary
Non-return to Zero Inverted (NRZI) 0 = no transition at beginning of interval 1 = transition at beginning of interval	
Bipolar – AMI 0 = no line signal 1 = positive or negative level, alternating for successive ones	
Pseudo-ternary 0 = positive or negative level, alternating for successive zeros.	
Manchester 0 = transition from high to low in middle of interval 1 = transition from low to high in middle of interval	
Differential Manchester Always a transition in middle of interval 0 = transition at beginning of interval 1 = no transition at beginning of interval	

Figure 2.6: Encoding schemes

Bipolar-AMI – In this scheme, a binary 0 is represented by no line signal, and a binary 1 is represented by a positive or negative pulse.

Pseudo-ternary – This scheme is opposite to bipolar-AMI, it is the binary 1 that is represented by the absence of a line signal, and the binary 0 by positive or negative pulses.

Manchester – In this scheme there is a transition at the middle of each bit period. The *mid-bit transition* serves as data and clocking mechanism. Binary 1 is represented by a low-to-high transition and binary 0 is represented by a high-to-low transition.

Differential Manchester – The mid-bit transition in this scheme is used only to provide clocking. The 0 is represented by the presence of a transition at the beginning of a bit period, and a 1 is represented by the absence of a transition at the beginning of a bit period.

2.3 FRAMING

In the physical layer, data transmission means transmitting bits in the form of a signal from the source to the destination. The physical layer has bit synchronization to ensure that the sender and receiver use the same bit durations and timing. The data link layer's task is to pack bits into frames, so that each frame is different from the other. For example; our postal system, where an envelope separates one piece of letter from the other. Here the envelope serves as the delimiter. Furthermore, the envelope identifies the sender and receiver addresses.

In the same manner *framing* in the data link layer differentiates one message from a particular source to a destination from some other message to other destinations. It adds a sender address and a destination address. The sender address helps the recipient acknowledge the arrival of the message and the destination address defines where the frame has to go. We can send the whole message in one frame.

However, since the flow and error control become inefficient with large frames, we divide a message into many small frames. If a large frame carries a whole message, even a single-bit error would require the retransmission of the entire message. But with small frames, a single-bit error affects only a small part of the entire message. Moreover, frames can be of fixed or variable size.

In *fixed-size framing* we need not define the boundaries of the frames. The fixed size of the frame acts as a delimiter. The ATM wide-area network is a good example of the usage of fixed size frames. In *variable-size framing* we need to define the end of the frame and the beginning of the next.

It is commonly used in local area networks. A *character-oriented approach* and *a bit-oriented approach* are two approaches used in variable-size framing.

2.3.1 Character-Oriented Protocol

The data carried in a character-oriented protocol are 8-bit characters from a coding system such as ASCII. The header includes the source and destination addresses and other control information. The trailer includes error detection or error correction redundant bits. All are multiples of 8 bits. To differentiate one frame from the other, a 1-byte (8 bit) flag is added at the beginning and the end of a frame. The figure below shows the format of a frame in a character-oriented protocol.

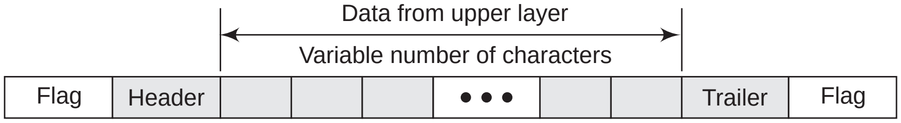


Figure 2.7(a): Frame format of character-oriented protocol

Initially we could only exchange text in character-oriented framing. But now it is used for other types of information such as graphics, audio, and video. Whatever the pattern may be, flag is the part of the information. In this case, when the receiver encounters this type of pattern in the middle of the data, assumes it has reached the end of the frame.

To avoid this problem, a *byte-stuffing* strategy was added to character-oriented framing, where a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. This extra byte is called the *escape character (ESC)* which has a predefined bit pattern. If the receiver comes across the ESC character, it removes it from the data section and takes the next character as data, not a delimiting flag.

Even this technique has some problem when the text contains one or more escape characters followed by a flag. The receiver rejects the escape character and keeps the flag. As a remedy, the escape characters that are part of the text must also be marked by another escape character. This is illustrated in the figure below. Today the universal coding systems use Unicode, which has 16-bit and 32-bit characters that conflict with 8-bit characters.

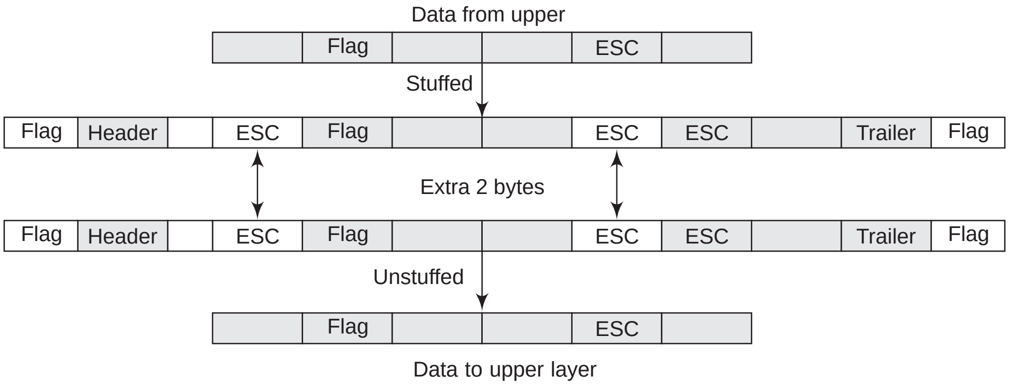
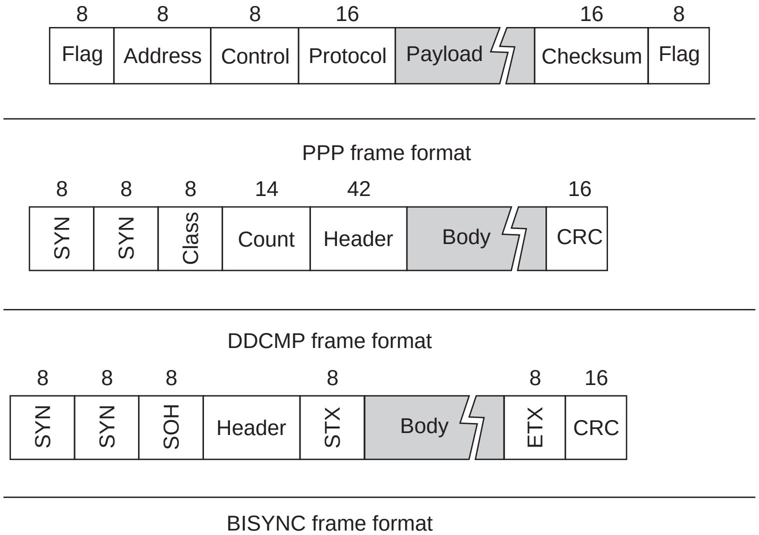


Figure 2.7(b): Byte Oriented Protocol



PPP frame format

DDCMP frame format

BISYNC frame format

Figure 2.7(c): Frame formats of byte oriented protocols

2.3.2 Bit-Oriented Protocols

In this method, the data section of a frame is a series of bits to be interpreted by the upper layer as text, graphic, audio, video, and other data. Here in addition to the header, we need a delimiter to separate one frame from the other. This protocol uses a special 8-bit pattern flag 01111110 as the delimiter to decide the beginning and the end of the frame. The figure below shows a frame in a bit-oriented protocol.

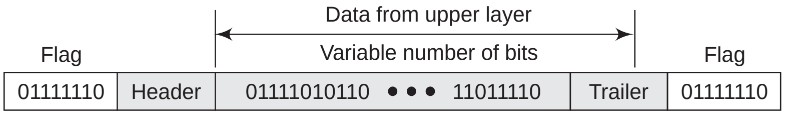


Figure 2.8: Bit oriented protocol

Unlike byte-oriented protocols, a bit-oriented protocol simply views the frame as a collection of bits. These bits could however be from any source; let's say, a character set from a text file, a set of pixels from an image or they may just be instructions to perform an executable statement.

The much popular HDLC (High-Level Data Link Control) protocol is an example of such a bit-oriented protocol. It has been derived from the SDLC (Synchronous Data Link Control) protocol developed by IBM. In the following discussion, we use HDLC as an example; its frame format is given in Figure 2.12.

HDLC uses a special sequence of bits; 01111110 to identify the beginning as well as the end of a frame. This sequence is also transmitted during any times that the link is idle so that the sender and receiver can keep their clocks synchronized.

In this way, both protocols essentially use the *sentinel* approach. Because this sequence can appear anywhere in the body of the frame; in fact, the bits 01111110 might cross byte boundaries, bit-oriented protocols use the analog form of the DLE character, a technique known as ***bit-stuffing***.

Bit-stuffing in the HDLC protocol works as follows. On the sending side, if anytime five consecutive 1s have been transmitted from the body of the message (i.e., excluding when the sender is trying to transmit the distinguished 01111110 sequence), the sender inserts a 0 before transmitting the next bit.

On the receiving side, if five consecutive 1s arrive, the receiver makes its decision based on the next bit it sees (i.e., the bit following the five 1s). If the next bit is a 0, it must have been stuffed, and so the receiver removes it. If the next bit is a 1, then one of two things is true: Either this is the end-of-frame marker or an error has been introduced into the bit stream.

By looking at the next bit, the receiver can distinguish between these two cases. If it sees a 0 (i.e., the last byte it has looked at is 01111110), then it is the end-of-frame marker; if it sees a 1 (i.e., the set of last eight bits it has looked at is 01111111), then there must have been an error and the whole frame is discarded. In the latter case, thereceiver has to wait for the next 01111110 before it can start receiving again, and as a consequence, there is the potential that the receiver will fail to receive two consecutive frames.

Obviously, there are still ways that framing errors can go undetected such as when an entire set of fake end-of-frame patterns is generated by errors, but such failures are relatively unlikely. Robust ways of detecting errors are discussed later.

An interesting characteristic of bit-stuffing, as well as character-stuffing, is that the size of a frame is dependent on the data that is being sent in the payload of the frame. It is in fact not possible to make all frames exactly the same size given that the data that might be carried in any frame is arbitrary.

2.3.3 Clock-based Framing: SONET

A third approach to framing is described by the Synchronous Optical Network (SONET) standard. For lack of a widely accepted generic term, we refer to this approach simply as ***clock-based* framing**.

SONET was first proposed by Bell Communications Research (Bellcore), and then developed under the American National Standards Institute (ANSI) for digital transmission over optical fiber; it has since been adopted by the ITU-T as the dominant standard for long-distance transmission of data over optical networks.

SONET addresses both the *framing* problem as well as the *encoding* problem. It also addresses a problem that is very important for phone companies—the multiplexing of several low-speed links onto one high-speed link.

There is a very slight resemblance between the previous framing schemes and SONET in that like all other schemes, a SONET frame too has somespecial information that tells the receiver where the frame starts and ends. However, there is no bit stuffing used; therefore, a frame's length does not depend on the data being sent.

An STS-1 frame is shown in Figure 2.9(a). It is arranged as nine rows of 90 bytes each and the first 3 bytes of each row are overhead, with the rest being available for data that is being transmitted over the link. The first 2 bytes of the frame contain a special bit pattern, and it is these bytes that enable the receiver to determine where the frame starts. However, since bit stuffing is not used, there is no reason why this pattern will not occasionally turn up in the payload portion of the frame.

To guard against this, the receiver looks for the special bit pattern consistently, hoping to see it appear once every 810 bytes, since each frame is 9 × 90 = 810 bytes long. When the special pattern turns up in the right place enough times, the receiver concludes that it is in *sync* and can then interpret the frame correctly.

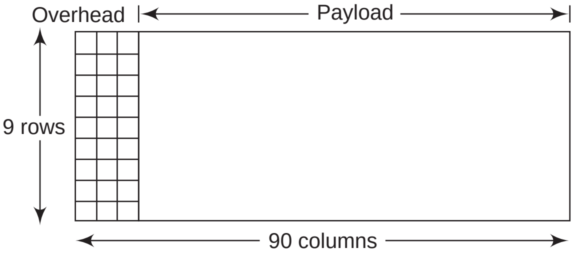


Figure 2.9(a): A SONET STS-1 frame

The overhead bytes of a SONET frame are encoded using NRZ, the simple encoding described in the previous section where 1s are high and 0s are low. However, to ensure that there are plenty of transitions to allow the receiver to recover the sender's clock, the payload bytes are scrambled. This is done by calculating the exclusive-OR (XOR) of the data to be transmitted and by the use of a well-known bit pattern.

The bitpattern, which is 127 bits long, has plenty of transitions from 1 to 0, so that XORing it with the transmitted data is likely to yield a signal with enough transitions to enable clock recovery. SONET supports the multiplexing of multiple low-speed links in the followingway. A given SONET link runs at one of a finite set of possible rates, ranging from51.84 Mbps (STS-1) to 2488.32 Mbps (STS-48) and beyond. (See Table 2.2 for the full set of SONET data rates.)

Note that all of these rates are integermultiples of STS-1. The significance for framing is that a single SONET frame cancontain sub-frames for multiple lower-rate channels. A second related feature is that each frame is 125 μs long. This means that at STS-1 rates, a SONET frame is 810 byteslong, while at STS-3 rates; each SONET frame is 2430 bytes long. Notice the synergy between these two features: 3 × 810 = 2430, meaning that three STS-1 frames fitexactly in a single STS-3 frame.

We can consider the STS-N frame to be a repetition of N STS-1 frames, with interleaving bytes; *i.e.*; a byte from the first frame is transmitted first, followed by a byte from the second frame, and so on. This is done to ensure that the bytes in each STS-1 frame are received at the other end at a constant rate of around 51 Mbps, rather than all of them being bundled up into one particular time frame. In other words, we could consider the STS-N signal as N concatenated STS-1 frames, denoted by STS-Nc.

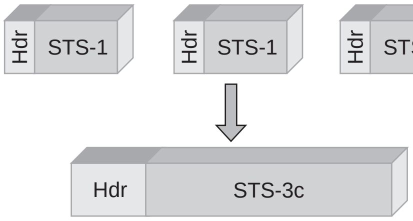


Figure 2.9(b): Three STS-1 frames multiplexed onto one STS-3c frame

Figure 2.9(b) depicts the concatenation of three STS-1 frames into a single STS-3c frame. The SONET link is referred to as STS-3c rather than STS-3 because an STS-3c payload indicates a single 155.25-Mbps pipe, whereas STS-3 should actually indicate three 51.84-Mbps links sharing a single fiber.

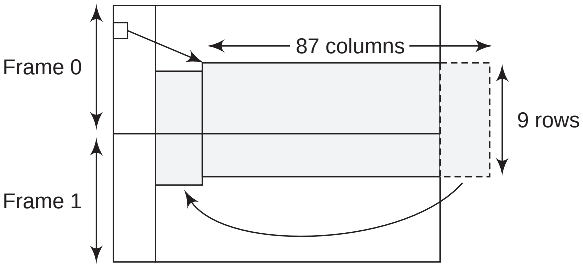


Figure 2.10: SONET frames out of phase

Finally, the preceding description of SONET is overly simplistic in that it assumes that the payload for each frame is completely contained within the frame. The STS-1 frame is simply a placeholder for the frame, where the actual payload may float across frame boundaries.

This situation is illustrated in Figure 2.10. Here we see both the STS-1 payload floating across two STS-1 frames, and the payload shifted some number of bytes to the right and, therefore, wrapped around. One of the fields in the frame overhead points to the beginning of the payload. The value of this capability is that it simplifies the task of synchronizing the clocks used throughout the carriers' networks, which issomething that carriers spend a lot of their time worrying about.

2.4 DETECTION AND CORRECTION OF ERRORS

The data link layer is responsible for ***detecting*** and ***handling*** **errors** that occur at the lower levels of the network stack.

Any time data can be corrupted during transmission from one node to another node. Many factors can change one or more bits of a message. We need some application to detect and correct these errors. Some data such as audio or video transmission may tolerate error for some extent but we expect high level of accuracy to transmit text. There are two types of errors. One is *single-bit error,* which means that only 1 bit of a given data is changed from 1 to 0 or from 0 to 1.

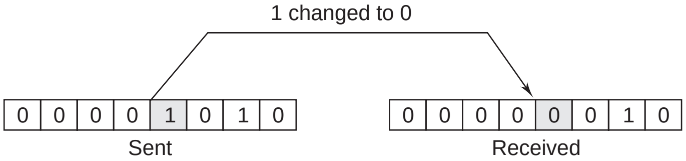


Figure 2.11(a): Single-bit error

The other error type is *burst error,* which means that 2 or more bits of a given data have changed from 1 to 0 0r from 0 to 1.

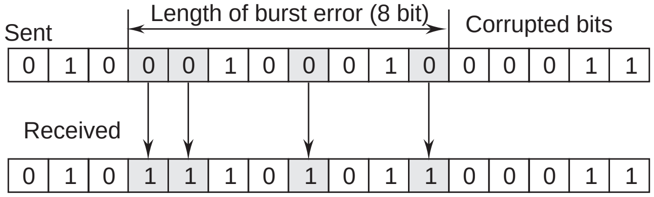


Figure 2.11(b): Burst error

The correction of errors is more difficult while comparing to error detection. The simple function of error detection is to find if any error has occurred and just inform yes or no. But in error correction we need to find exact number of bits that are corrupted and their location in the message.

Redundancy – The main concept of detecting or correcting error is based on *redundancy.* An *extra* bit is usually sent with the data to be able to detect or correct errors. These redundant bits are added by the sender and removed by the receiver.

Types of error correction – There are two main types of error correction available; *Forward error correction* and *retransmission.* In the forward error correction process, the receiver tries to guess the message by identifying the redundant bits. This method is applicable if the number of errors is small. In retransmission, the receiver asks the sender to resend the message when it detects the error,

Coding – Redundancy can be achieved by using different coding schemes. The sender adds redundant bits through a process, which creates synchronization between the redundant bits and the actual data bits. The receiver checks the connection between the two sets of bits to detect or correct the errors. The figure below explains the structure of encoder and decoder. The two categories of coding are *block coding* and *convolution coding.* In the following section we look into block coding. Convolution coding is more complex.

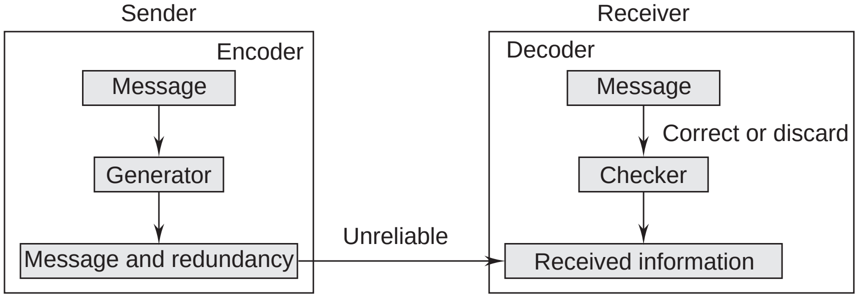


Figure 2.12: Block coding for redundancy

Block Coding – In block coding, the message is divided into blocks. In this method each of *k* bit is called *datawords.* Redundant bit *r* is added to each block, then the length *n* = k + r. The result of *n*-bit blocks is called *codewords.* This method has a set of datawords, each of size *k,* and set of codewords, each of size of *n.* With *k bits,* combination of 2^k datawords can be created and with *n* bits, a combination of 2^n codewords can be created. Moreover, as n > k, the number of possible codewords is more than the number of possible datawords.

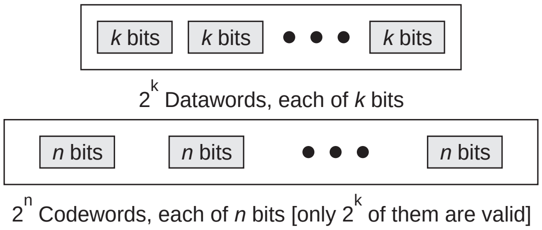

k bits
k bits
• • •
k bits
2^k Datawords, each of k bits
n bits
n bits
• • •
n bits
2^n Codewords, each of n bits [only 2^k of them are valid]

Figure 2.13: Creation of datawords and codewords

(*a*) **Error Detection in Block Coding:** To detect the change in original codeword, the receiver should have a list of valid codewords and the original codeword must have changed to an invalid one. The sender produces codewords out of datawords by using a generator, which follows the rules and procedures of encoding. Each codeword sent to the receiver may get altered during transmission. If the received codeword is the same as one of the valid code words, then the word is accepted and the related dataword is removed for use. Sometimes though the codeword is corrupted during transmission, the received word still matches a valid codeword. In this case the error remains undetected. This type of coding detects only one error. Therefore, an error-detecting code can detect only the types of errors for which it is designed and the other types of errors may remain unnoticed.

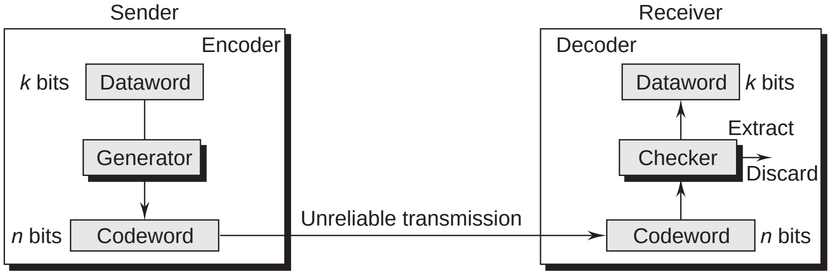


Figure 2.14(a): Error detection in block coding

(*b*) **Error Correction in Block Coding:** The task of error detection is to find that the received codeword is invalid. In error correction the receiver has to find the original codeword sent. More redundant bits are needed for error correction than for error detection. The figure below shows the error correction in block coding.

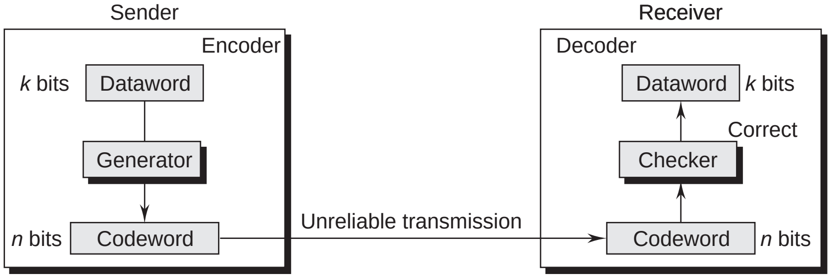


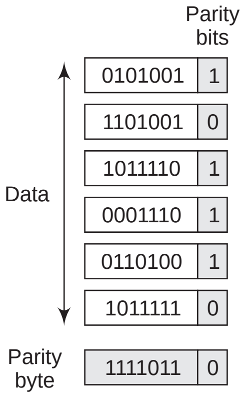
Figure 2.14(b): Error correction in block coding

2.4.1 One-Dimensional Parity

It is based on "simple" (one-dimensional) parity, which usually involves adding one extra bit to a 7-bit code to balance the number of 1s in the byte. For example, odd parity sets the eighth bit to 1 if needed to give an odd number of 1s in the byte, and even parity sets the eighth bit to 1 if needed to give an even number of 1s in the byte.

Two-dimensional parity does a similar calculation for each bit position across each of the bytes contained in the frame. This results in an extra parity byte for the entire frame, in addition to a parity bit for each byte.

Figure 2.15 illustrates how two-dimensional even parity works for an example frame containing 6 bytes of data. Notice that the third bit of the parity byte is 1 since there are an odd number of 1s in the third bit across the 6 bytes in the frame. It can be shown that two-dimensional parity catches all 1-, 2-, and 3-bit errors, and most 4-bit errors. Here we have added 14 bits of redundant information to a 42 bit message.



		Parity bits
Data	0101001	1
	1101001	0
	1011110	1
	0001110	1
	0110100	1
	1011111	0
Parity byte	1111011	0

Figure 2.15: Two-dimensional parity

2.4.2 Internet Checksum Algorithm

A second approach to error detection is demonstrated by the Internet checksum. The idea behind the Internet checksum is very simple—you add up all the words that are transmitted and then transmit the result of that sum. The result is called the checksum. The receiver performs the same calculation on the received data and compares the result with the received checksum. If any transmitted data, including the checksum itself, is corrupted, then the results will not match, so the receiver knows that an error occurred.

You can imagine many different variations on the basic idea of a checksum. The exact scheme used by the Internet protocols works as follows:

Consider the data being *check-summed* as a sequence of 16-bit integers. Add them together using 16-bit one's complement arithmetic (explained below) and then take the one's complement of the result.

Consider, for example, the addition of "5" and "3" in one's complement arithmetic on 4-bit integers.

5 means 1010

3 means 1100

Adding 1010 and 1100 gives us 0110, ignoring the carry.

In one's complement arithmetic, a carry from the most significant bit causes us to increment the result, giving 0111, which is the one's complement representation of "8" (obtained by inverting the bits in 1000).

2.4.3 Cyclic Redundancy Check

The Cyclic redundancy check mechanism uses a powerful mathematical approach for error detection.

Usually, a 32-bit CRC can be said to give enough protection against common bit errors in messages even up to thousands of bytes long.

The message is represented by a polynomial, where the value of each bit in the message is used as the coefficient for the corresponding term in the polynomial, with the most significant bit representing the highest-order term.

For example, an 8-bit message consisting of the bits 10011010 corresponds to the polynomial

$$\begin{aligned} \mathrm{M}(x) &= 1 \times x^7 + 0 \times x^6 + 0 \times x^5 + 1 \times x^4 + 1 \times x^3 + 0 \times x^2 + 1 \times x^1 + 0 \times x^0 \\ &= x^7 + x^4 + x^3 + x^1 \end{aligned}$$

For the purpose of calculating a CRC, a sender and receiver have to agree on a divisor polynomial, C(x), of degree k.

For example, suppose

$$\mathrm{C}(x) = x^3 + x^2 + 1.$$

In this case, $k = 3$.

The choice of C(x) has a significant impact on what types of errors can be reliably detected. There are several divisor polynomials that are very good choices for various environments and the exact choice is normally made as part of theprotocol design.

For example, the Ethernet standard uses a well-known polynomial of degree 32.

When a sender wishes to transmit a message M(x) that is $n + 1$ bits long, whatis actually sent is the $(n + 1)$-bit message plus k bits.

Let P(x), be a polynomial used to denote the entire message to be transmitted, including the redundant bits. Now, we must try and make P(x) exactly divisible by C(x). If P(x) is transmitted over a link and there are no errors introduced during transmission, then the receiver should be able to divide P(x) by C(x) exactly, leaving a remainder of zero.

On the otherhand, if some error is introduced into P(x) during transmission, then in most cases the received polynomial will no longer be exactly divisible by C(x), and thus the receiver will obtain a non-zero remainder, implying that an error has occurred.

For example, the polynomial $x^3 +1$ can be divided by $x^3 + x^2 +1$ (because they are both of degree 3) and the remainder would be $0 \times x^3 + 1 \times x^2 + 0 \times x^1 + 0 \times x^0 = x^2$(obtained by XORing the coefficients of each term). In terms of messages, we could say that 1001 can be divided by 1101 and leaves a remainder of 0100. The remainder is just the bitwise exclusive-OR of the two messages.

Nowthat we know the basic rules for dividing polynomials, we are able to do long from the original message M(x), is k bits longer than M(x), and is exactly divisible by C(x). We can do this in the following way:

1. Multiply M(x) by x^k; that is, add k zeroes at the end of the message. Call this Zero-extended message T(x).
2. Divide T(x) by C(x) and find the remainder.
3. Subtract the remainder from T(x).

It should be obvious that what is left at this point is a message that is exactlydivisible by C(x). We may also note that the resulting message consists of M(x) followed by the remainder obtained in step 2, because when we subtracted the remainder(which can be no more than k bits long), we were just XORing it with the k zeroesadded in step 1.

Consider the message $x^7 + x^4 + x^3 + x^1$, or 10011010. We begin by multiplyingby x^3, since our divisor polynomial is of degree 3. This gives 10011010000.We dividethis by C(x), which corresponds to 1101 in this case. Figure 2.17 shows the polynomiallong division operation. Given the rules of polynomial arithmetic described above, thelong division operation proceeds much as it would if we were dividing integers. Thusin the first step of our example, we see that the divisor 1101 divides once into the first four bits of the message (1001), since they are of the same degree, and leavesa remainder of 100 (1101 XOR 1001).

The next step is to bring down a digit from the message polynomial until we get another polynomial with the same degree as C(x), in this case 1001.We calculate the remainder again (100) and continue until the calculation is complete. Note that the "result" of the long division, which appears at the top of the calculation, is not really of much interest—it is the remainder at the end that matters.

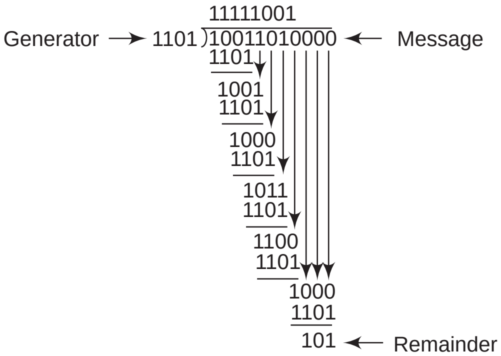


Figure 2.16(a): CRC calculation using polynomial long division

We actually send 1001101010; this turns out to be just the original message with the remainder from the long division calculation appended to it.

The recipient divides the received polynomial by C(x) and, if the result is 0, concludes that there were no errors. If the result is nonzero, it may be necessary to discard the message containing the error; with some codes, it may be possible to correct a small error (*e.g*., if the error affected only one bit). A code that enables error correction is called an error-correcting code (ECC).

Now we will consider the question of where the polynomial C(x) comes from. Intuitively, the idea is to select this polynomial so that it is very unlikely to divide evenly into a message that has errors introduced into it. If the transmitted message is P(x), we may think of the introduction of errors as the addition of another polynomial E(x), so the recipient sees P(x) + E(x). The only way that an error could slip by undetected would be if the received message could be evenly divided by C(x), and since we know that P(x) can be evenly divided by C(x), this could only happen if E(x) can be divided evenly by C(x). The trick is to pick C(x) so that this is very unlikely for common types of errors.

One common type of error is a single-bit error, which can be expressed as E(x) = x^i when it affects bit position ***i***. If we select C(x) such that the first and the last term are non-zero, then we already have a two-term polynomial that cannot divide evenly into the one term E(x). Such a C(x) can, therefore, detect all single-bit errors. In general, it is possible to prove that the following types of errors can be detected by a C(x) with the stated properties:

All single-bit errors, as long as the x^k and x^0 terms have nonzero coefficients.

- All double-bit errors, as long as C(x) has a factor with at least three terms.
- Any odd number of errors, as long as C(x) contains the factor (x + 1).
- Any "burst" error (*i.e*., sequence of consecutive-*errored* bits) for which the length of the burst is less than k-bits. (Most burst errors of larger than k-bits can also be detected.)

Six versions of C(x) are widely used in link-level protocols (shown in Table 2.4).

Table 2.4: Common CRC polynomials

CRC	**C(x)**
CRC-8	$x^8 + x^2 + x^1 + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7$ $+ x^5 + x^4 + x^2 + x + 1$

For example, the Ethernet and 802.5 networks described later in this chapter use CRC-32, while HDLC uses CRC-CCITT. ATM, as described in Chapter 3, uses CRC-8, CRC-10, and CRC-32. Finally, we note that the CRC algorithm, while seemingly complex, is easily implemented in hardware using a k-bit shift register and XOR gates. The number of bits in the shift register equals the degree of the generator polynomial (k). Figure 2.16 shows the hardware that would be used for the generator $\boldsymbol{x^3 + x^2 + 1}$ from our previous example.

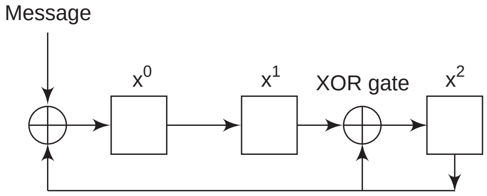


Figure 2.16(b): CRC calculation using shift register

The message is shifted in from the left, beginning with the most significant bit and ending with the string of k zeroes that is attached to the message, just as in the long division example. When all the bits have been shifted in and appropriately XORed, the register contains the remainder; that is, the CRC (most significant bit on the right). The position of the XOR gates is determined as follows:

If the bits in the shift register are labelled 0 through k – 1, left to right, then put an XOR gate in front of bit n if there is a term x^n in the generator polynomial. Thus, we see an XOR gate in front of positions 0 and 2 for the generator $\boldsymbol{x^3 + x^2 + x^0}$.

2.5 RELIABLE TRANSMISSION

Frames are sometimes corrupted during transmission. Some error codes are strong enough also to correct errors but the overhead is typically too large to handle the range of bit and burst errors. Even after using error correcting codes also some errors will be too severe to be corrected. As a result, some corrupt frames must be discarded.

A link-level protocol that wants to deliver frames reliably must somehow recover from these discarded (lost) frames. This is usually accomplished using a combination of two fundamental mechanisms; **Acknowledgements** and **timeouts.**

An ***acknowledgement*** (ACK for short) is a small ***control frame*** that a protocol sends back to its peer saying that it has received an earlier frame. By control frame we mean a header without any data, although a protocol can piggyback an ACK on a data frame, it just happens to be sending in the opposite direction.

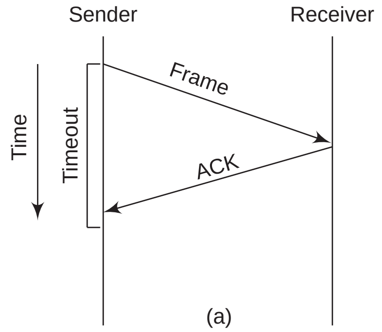
The receipt of an acknowledgment indicates to the sender of the original frame that its frame was successfully delivered. If the sender does not receive an acknowledgement after a reasonable amount of time, then it retransmits the original frame. This action of waiting a reasonable amount of time is called a ***timeout***.

The general strategy of using acknowledgements and timeouts to implement reliable delivery is sometimes called *automatic repeat request* (normally abbreviated ARQ). This section describes three different ARQ algorithms using generic language.

2.5.1 Stop-and-Wait

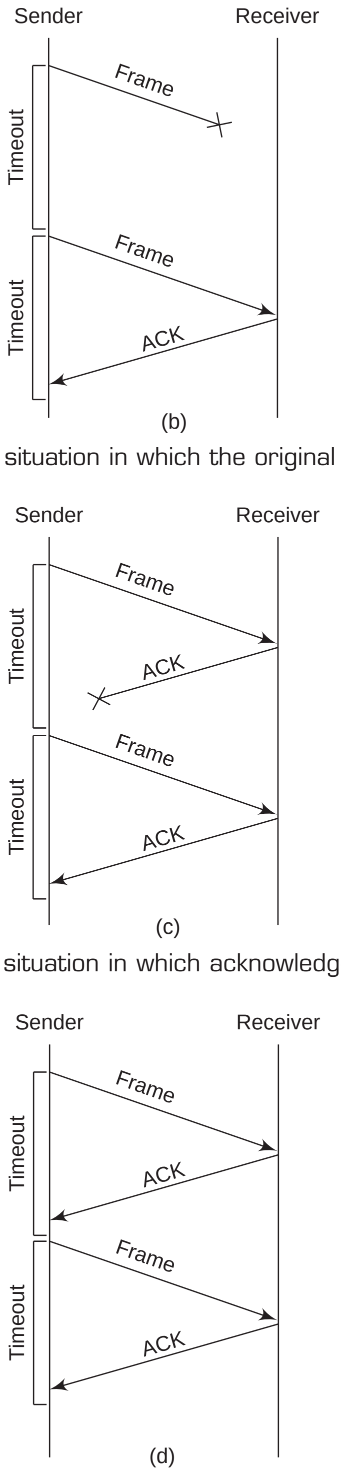
The simplest ARQ scheme is the stop-and-wait algorithm. The idea of stop-and-waitis straightforward: After transmitting oneframe, the sender waits for an acknowledgement before transmitting the next frame. If the acknowledgement does not arrive after acertain period of time, the sender times outand retransmits the original frame. Figure 2.17 illustrates four different scenarios that result from this basic algorithm. This figure is a timeline, a common way to depict a protocol's behavior.

The sending side is represented on the left, the receiving side is depicted on the right, and time flows from top to bottom.



(a)

(a) shows the situation, in which the ACK is received before the timer expires,



(b) Shows the situation in which the original frame is lost

(c) Shows the situation in which acknowledgement is lost

(d) Shows the situation in which the timeout fires too soon.

Figure 2.17: Stop-and-wait algorithm

By "lost" we mean that the frame was corrupted while in transit, that this corruption was detected by an error code on the receiver, and that the frame was subsequently discarded.

There is one important nuance in the **stop-and-wait** algorithm. Suppose the sender sends a frame and the receiver acknowledges it, but the acknowledgement is either lost or delayed in arriving. In that case the sender times out and retransmits the original frame, but the receiver will think that it is the next frame, since it correctly received and acknowledged the first frame. This has the potential to cause duplicate copies of a frame to be delivered.

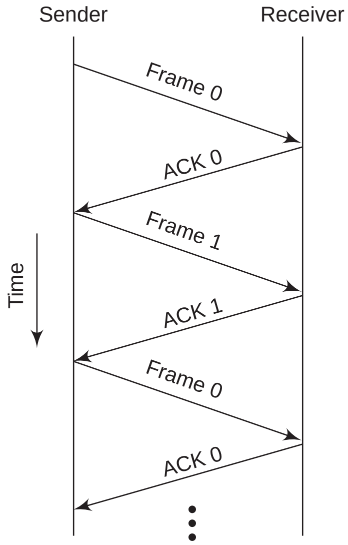


Figure 2.18: Timeline for stop-and-wait with 1-bit sequence number

To address this problem, the header for a stop-and-wait protocol usually includes a 1-bit sequence number—that is, the sequence number can take on the values 0 and 1—and the sequence numbers used for each frame alternate, as illustrated in Figure 2.18. Thus, when the sender retransmits frame 0, the receiver can determine that it is seeing a second copy of frame 0 rather than the first copy of frame 1 and therefore can ignore it (the receiver still acknowledges it, in case the first ACK was lost).

The main shortcoming of the stop-and-wait algorithm is that it allows the sender to have only one outstanding frame on the link at a time, and this may be far below the link's capacity.

Consider, for example, a 1.5-Mbps link with a 45-ms round-trip time. This link has a delay × bandwidth product of 67.5 Kb, or approximately 8 KB.

Since the sender can send only one frame per RTT, and assuming a frame size of 1 KB, this implies a maximum sending rate of

Bits per Frame ÷ Time per Frame

$$= 1024 \times 8 \div 0.045$$
$$= 182 \text{ Kbps}.$$

This amounts to just one-eighth of the link's capacity. To use the link fully, then, we'd like the sender to be able to transmit up to eight frames before having to wait for an acknowledgement, which is leading to another algorithm knows as sliding window algorithm.

2.5.2 Sliding Window

Consider again the scenario in which the link has a delay ×bandwidth product of 8 KB and frames are of 1-KB size. We would like the sender to be ready to transmit the ninth frame at pretty much the same moment that the ACK for the first frame arrives. The algorithm that allows multiple bytes or packets to be sent before waiting for an acknowledgement, and an illustrative timeline is given in Figure 2.19.

The sliding window algorithm works as follows.

First, the sender assigns a ***sequence number,*** denoted SeqNum, to each frame.

The sender maintains three variables:

- The *send window size*, denoted SWS, gives the upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit;
- LAR denotes the sequence number of the *last acknowledgement received*;
- LFS denotes the sequence number of the *last frame sent*.

The sender also maintains the following invariant:

$$\text{LFS} - \text{LAR} \le \text{SWS}.$$

When an acknowledgement arrives, the sender moves LAR to the right, thereby allowing the sender to transmit another frame. Also, the sender associates a timer with each frame it transmits, and it retransmits the frame should the timer expire before an ACK is received.

Notice that the sender has to be willing to buffer up to SWS frames since it must be prepared to retransmit them until they are acknowledged.

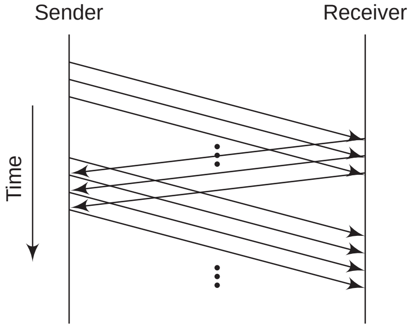


Figure 2.19: Timeline for the sliding window algorithm

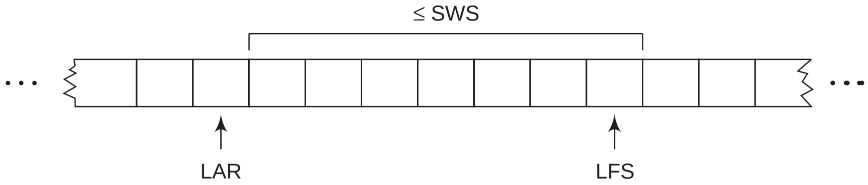


Figure 2.20(a): Sliding window on sender

The receiver maintains the following three variables:

- The *receive window size*, denoted RWS, gives the upper bound on the number of out-of-order frames that the receiver is willing to accept;
- LAF denotes the sequence number of the *largest acceptableframe*;
- LFR denotes the sequence number of the *last frame received*.

The receiver also maintains the following invariant:

LAF –LFR ≤RWS

This situation is illustrated in Figure 2.20(b).

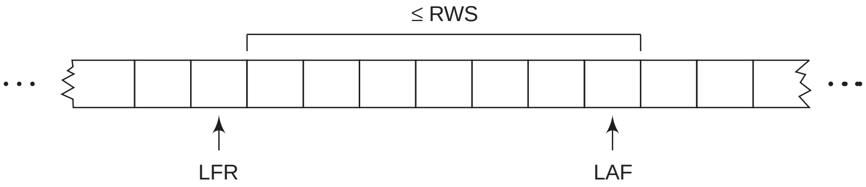


Figure 2.20(b): Sliding window on receiver

When a frame with sequence number SeqNum arrives, the receiver takes the following action:

If SeqNum ≤ LFR or SeqNum > LAF, then the frame is outside the receiver’s window and it is discarded.

If LFR < SeqNum ≤ LAF, then the frame is within the receiver’s window and it is accepted.

Now the receiver needs to decide whether or not to send an ACK.

Let SeqNumToAck denote the largest sequence number not yet acknowledged, such that all frames with sequence numbers less than or equal to SeqNumToAck have been received. The receiver acknowledges the receipt of SeqNumToAck, even if higher-numbered packets have been received.

This acknowledgment is said to be cumulative. It then sets LFR = SeqNumToAck and adjusts

LAF = LFR + RWS.

For example, suppose LFR = 5 (*i.e.*, the last ACK the receiver sent was for sequence number 5), and RWS = 4. This implies that LAF = 9 .If 7 and 8 arrive, they will be buffered because they are within the receiver’s window.

However, no ACK needs to be sent since frame 6 is yet to arrive. Frames 7 and 8 are said to have arrived out of order. (Technically, the receiver could resend an ACK for frame 5, when frames 7 and 8 arrive.)

If frame 6 then arrives—perhaps it is late because it was lost the first time and had to be retransmitted, or perhaps it was simply delayed—the receiver acknowledges frame 8, bumps LFR to 8, and sets LAF to 12. If frame 6 was in fact lost, then a timeout will have occurred at the sender, making it to retransmit frame 6.

We observe that when a timeout occurs, the amount of data in transit decreases, since the sender is unable to advance its window until frame 6 is acknowledged. This means that when packet losses occur, this scheme is no longer keeping the pipe full.

The longer it takes to notice that a packet loss has occurred, the more severe this problem becomes.

Notice that in this example, the receiver could have sent a *negative acknowledgment* (NAK) for frame 6 as soon as frame 7 arrived.

However, this is unnecessary since the sender's timeout mechanism is sufficient to catch this situation, and sending NAKs adds additional complexity to the receiver. Also, as we mentioned, it would have been authentic to send additional acknowledgments of frame 5 when frames 7 and 8 arrived; in some cases, a sender can use duplicate ACKs as a clue that a frame was lost.

Both approaches help to improve performance by allowing early detection of packet losses. Yet another variation on this scheme would be to use *selective acknowledgments*. That is, the receiver could acknowledge exactly those frames it has received, rather than just the highest-numbered frame received in order. So, in the above example, thereceiver could acknowledge the receipt of frames 7 and 8.

Giving more information to the sender makes it potentially easier for the sender to keep the pipe full, but adds complexity to the implementation. The sending window size is selected according to how many frames we want to have outstanding on the link at a given time; SWS is easy to compute for a given ***delay × bandwidth*** product.

On the other hand, the receiver can set RWS to whatever it wants. Two common settings are RWS = 1, which implies that the receiver will not buffer anyframes that arrive out of order, and RWS = SWS, which implies that the receiver can buffer any of the frames the sender transmits. It makes no sense to set RWS >SWS since it's impossible for more than SWS frames to arrive out of order.

Finite Sequence Numbers and Sliding Window

We now return to the one simplification we introduced into the algorithm — our assumption that sequence numbers can grow infinitely large. In practice, of course, a frame's sequence number is specified in a header field of some finite size.

For example, a 3-bit field means that there are eight possible sequence numbers, 0 . . . 7. This makes it necessary to reuse sequence numbers or, stated another way, sequence numbers wrap around.

This introduces the problem of being able to distinguish between different incarnations of the same sequence numbers, which implies that the number of possible sequence numbers must be larger than the number of outstanding frames allowed. For example, ***stop-and-wait*** allowed one outstanding frame at a time and had two distinct sequence numbers.

Suppose we have one more number in our space of sequence numbers than we have potentially outstanding frames; that is, SWS ≤ MaxSeqNum – 1, where MaxSeqNum is the number of available sequence numbers. Is this sufficient?

The answer depends on RWR. If RWS = 1, then MaxSeqNum ≥ SWS + 1 is sufficient. If RWS is equal to SWS, then having a MaxSeqNum just one greater than the sending window size is not good enough.

To see this, consider the situation in which we have the eight sequence numbers 0 through 7, and SWS = RWS = 7. Suppose the sender transmits frames 0-6, they are successfully received, but the ACKs are lost. The receiver is now expecting frames 7, 0-5, but the sender times out and sends frames 0-6.

Unfortunately, the receiver is expecting the second form of frames 0-5, but gets the first form of these frames. This is exactly the situation we wanted to avoid. It turns out that the sending window size can be no more than half as big as the number of available sequence numbers when RWS = SWS, or stated more precisely, SWS < (MaxSeqNum+ 1)/2

Intuitively, what this means is that the *sliding window* protocol alternates between the two halves of the sequence number space, just as *stop-and-wait* alternates between sequence numbers 0 and 1. The only difference is that it continually slides between the two halves rather than discretely alternating between them.

Note that this rule is specific to the situation where RWS = SWS. Also note that the relationship between the window size and the sequence number space depends on an assumption that is quite easy to overlook, namely, that frames are not reordered in transit. This cannot happen on a direct point-to-point link since there is no way for one frame to overtake another during transmission.

The **sliding window protocol** is perhaps the best-known algorithm in computer networking. What can be confusing about the algorithm, however, is that it can serve three different roles.

The first role which is also the core function of the algorithm is to reliably deliver frames across an unreliable link.

The second role that the sliding window algorithm can serve is to preserve the order in which frames are transmitted. This is easy to do at the receiver, since each frame has a sequence number. The receiver just makes sure that it does not pass a frame up to the next-higher-level protocol until it has already passed up all frames with a smaller sequence number.

That is, the receiver buffer does not pass along out-of-order frames. The version of the sliding window algorithm described in this section also preserves frame order, although we could imagine a variation in which the receiver passes frames to the next protocol without waiting for all earlier frames to be delivered.

The third role that the sliding window algorithm sometimes plays is to support *flow control*—a feedback mechanism by which the receiver is able to check the sender. Such a mechanism is used to keep the sender from overrunning the receiver, that is, from transmitting more data than the receiver is able to process.

This is usually accomplished by augmenting the sliding window protocol so that the receiver not only acknowledges the frame it has received, but also informs the sender of how many frames it has room to receive. The number of frames that the receiver is capable of receiving corresponds to how much free buffer space it has. As in the case of ordered delivery, we need to make sure that flow control is necessary at the link level before incorporating it into the sliding window protocol.

2.6 CONCURRENT LOGICAL CHANNELS

The data link protocol used in the ARPANET provides an interesting alternative to the sliding window protocol, in that it is able to keep the pipe full while still using the simple stop-and-wait algorithm.

One important consequence of this approach is that the frames sent over a given link are not kept in any particular order. The protocol also does not imply anything about flow control.

The idea underlying the ARPANET protocol, which we refer to as *concurrent logical channels*, is to multiplex several logical channels onto a single point-to-pointlink and to run the stop-and-wait algorithm on each of these logical channels.

There is no relationship maintained among the frames sent on any of the logical channels, yet because a different frame can be outstanding on each of the several logical channels, the sender can keep the link full.

The sender keeps 3 bits of state for each channel: A Boolean, saying whether the channel is currently busy; the 1-bit sequence number to use the next time a frame is sent on this logical channel; and the next sequence number to expect on a frame that arrives on this channel.

When the node has a frame to send, it uses the lowest idle channel, and otherwise it behaves just like *stop-and-wait*.

In practice, the ARPANET supported 8 logical channels over each ground link and 16 over each satellite link. In the ground-link case, the header for each frame included a 3-bit channel number and a 1-bit sequence number, for a total of 4 bits.

This is exactly the number of bits the sliding window protocol requires to support up to eight outstanding frames on the link when RWS = SWS.

2.6.1 Ethernet (802.3)

The Ethernet is easily the most successful local area networking technology of the last 20 years. Developed in the mid-1970s by researchers at the Xerox Palo Alto Research Center (PARC), the Ethernet is a working example of the more general Carrier Sense Multiple Access with Collision Detect (CSMA/CD) local area network technology.

The CSMA (Carrier Sense Multiple Access) method is developed to reduce the chance of collision hence increase the performance. If a station senses the medium before using it, the chances of collision can be reduced. CSMA can reduce the chances of collision but it cannot eliminate it. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) increases the algorithm to handle the collision. In CSMA/CD method a station monitors the medium after it sends a frame to see if the transmission was successful. If there is a collision, the frame is sent again.

Digital Equipment Corporation and Intel Corporation joined Xerox to define a 10-Mbps Ethernet standard in 1978. This standard then formed the basis for IEEE standard 802.3. The standard also defines a much wider collection of physical media over which Ethernet can operate, and more recently, it has been extended to include a 100-Mbps version called Fast Ethernet and a 1000-Mbps version called Gigabit Ethernet.

We mainly focus on 10-Mbps Ethernet, since it is typically used in multiple-access mode.

Physical Properties

An Ethernet segment is implemented on a coaxial cable of up to 500 m, similar to that used for cable TV. Hosts connect to an Ethernet segment by tapping into it; taps must be at least 2.5 m apart. A small device, called transceiver, is directly attached to the tap that detects when the line is idle and drives the signal when the host is transmitting. It also receives incoming signals. The transceiver is, in turn, connected to an Ethernet adaptor, which is plugged into the host. All the logic that makes up the Ethernet protocol is implemented in the adaptor. This configuration is shown in Figure 2.21(a).

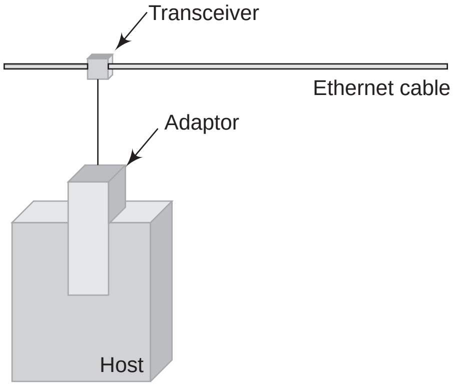


Figure 2.21(a): Ethernet transceiver and adaptor

Multiple Ethernet segments can be joined together by **repeaters**. A *repeater* is a device that forwards digital signals, much like an amplifier forwards analog signals. However, no more than four repeaters may be positioned between any pair of hosts, meaning that an Ethernet has a total reach of only 2500m. For example, using just two repeaters between any pair of hosts, supports a configuration similar to the one illustrated in Figure 2.21(b), that is, a segment running down the centre of a building with a segment on each floor. Moreover, an Ethernet is limited to supporting a maximum of 1024 hosts.

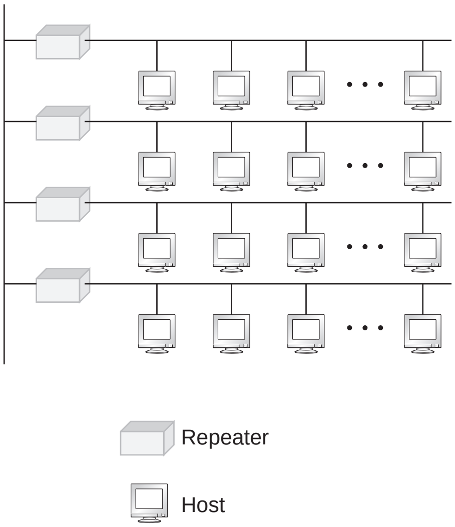


Figure 2.21(b): Ethernet repeater

Any signal placed on the Ethernet by a host is broadcast over the entire network; that is, the signal is propagated in both directions, and repeaters forward the signal on all outgoing segments. Terminators attached to the end of each segment absorb the signal and keep it from bouncing back and interfering with trailing signals. The Ethernet uses the Manchester encoding scheme.

Ethernet can be constructed from a thinner cable known as 10Base2; the original cable is called 10Base5 (the two cables are commonly called *thin-net* and *thick-net*, respectively). The “10” in 10Base2 means that the network operates at 10 Mbps, “Base” refers to the fact that the cable is used in a baseband system, and the “2” means that a given segment can be no longer than 200 m (a segment of the original 10Base5 cable can be up to 500 m long).

A third cable technology is predominantly used, called 10BaseT, where the "T" stands for twisted pair. Typically, Category 5 twisted pair wiring is used. It provides a data rate of 100 megabits per second. Both 100-Mbps and 1000-Mbps Ethernets also run over Categorytwisted pair, up to distances of 100 m. 100BASE-T is based on the older Ethernet standard. Since it is 10 times faster than Ethernet, it is often referred to as Fast Ethernet.

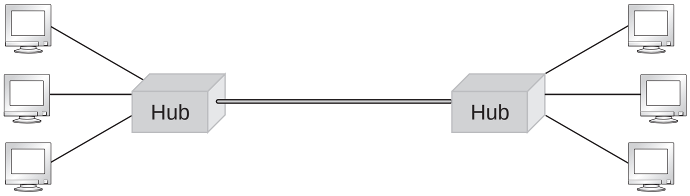


Figure 2.21(c): Ethernet hub

Because the cable is so thin, you do not tap into a 10Base2 or 10BaseT cable in the same way as you would with 10Base5 cable. With 10Base2, a T-joint is spliced into the cable. In effect, 10Base2 is used to daisy-chain a set of hosts together. With 10BaseT, the common configuration is to have several point-to-point segments coming out of a multi-way repeater, sometimes called a *hub*, as illustrated in Figure 2.21(c).

Multiple 100-Mbps Ethernet segments can also be connected by a hub, but the same is not true of 1000-Mbps segments.

It is important to understand that whether a given Ethernet spans a single segment, a linear sequence of segments connected by repeaters, or multiple segments connected in a star configuration by a hub, data transmitted by any one host on that Ethernet reaches all the other hosts. This is the good news. The bad news is that all these hosts are competing for access to the same link, and as a consequence, they are said to be in the same *collision domain*.

Access Protocol

Let us now see the algorithm that controls access to the shared Ethernet link. This algorithm is commonly called the Ethernet's *media access control* (MAC) and is typically implemented in hardware on the network adaptor.

First, however, we describe the Ethernet's frame format and addresses.

Frame Format

Each Ethernet frame is defined by the format given in Figure 2.22.

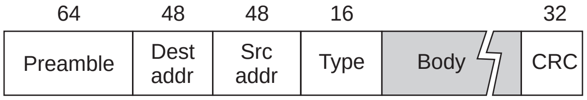


Figure 2.22: Ethernet frame format

The 64-bit preamble allows the receiver to synchronize with the signal; it is a sequence of alternating 0s and 1s. Both the source and destination hosts are identified with a 48-bit address.

The packet type field serves as the de-multiplexing key; that is, it identifies to which of possibly many higher-level protocols this frame should be delivered. Each frame contains up to 1500 bytes of data. Minimally, a frame must contain at least 46 bytes of data, even if this means the host has to pad the frame before transmitting it.

The reason for this minimum frame size is that the frame must be long enough to detect a collision. Finally each frame includes a 32-bit CRC.

Ethernet is a bit-oriented framing protocol. An Ethernet frame has a 14-byte header; two 6-byte addresses and a 2-byte type field. The sending adaptor attaches the preamble, CRC, and postamble before transmitting, and the receiving adaptor removes them. This type field is the first thing in the data portion of the 802.3 frames; that is, it immediately follows the 802.3 header.

Fortunately, since the Ethernet standard has avoided using any type values less than 1500 (the maximum length found in an 802.3 header), and the type and length fields are in the same location in the header, it is possible for a single device to accept both formats, and for the device driver running on the host to interpret the last 16 bits of the header as either a type or a length.

Addresses

Each host on an Ethernet—in fact, every Ethernet host in the world—has a unique Ethernet address. Technically, the address belongs to the adaptor, not the host; it is usually burned into ROM. Ethernet addresses are typically printed in a form humans can read as a sequence of six numbers separated by colons. Each number corresponds to 1 byte of the 6-byte address and is given by a pair of hexadecimal digits, one for each of the 4-bit nibbles in the byte; leading 0s are dropped.

For example, **8:0:2b:e4:b1:2** is the human-readable representation of Ethernet address 00001000 00000000 00101011 11100100 10110001 00000010

To ensure that every adaptor gets a unique address, each manufacturer of Ethernet devices is allocated a different prefix that must be prepended to the address on every adaptor they build. For example, Advanced Micro Devices (AMD) has been assigned the 24-bit prefix x080020 (or **8:0:20**).

A given manufacturer then makes sure the address suffixes it produces are unique.

Each frame transmitted on an Ethernet is received by every adaptor connected to that Ethernet. Each adaptor recognizes those frames addressed to its address and passes only those frames on to the host. (An adaptor can also be programmed to run in *promiscuous* mode, in which case it delivers all received frames to the host, but this is not the normal mode).

In addition to these *unicast* addresses, an Ethernet address consisting of all 1s is treated as a *broadcast* address; all adaptors pass frames addressed to the broadcast address up to the host. Similarly, an address that has the first bit set to 1 but is not the broadcast address is called a *multicast* address. A given host can program its adaptor to accept some set of multicast addresses. Multicast addresses are used to send messages to some subset of the hosts on an Ethernet (e.g., all file servers).

To summarize, an Ethernet adaptor receives all frames and acceptsframes addressed to its own address, frames addressed to the broadcast address, frames addressed to a multicast address, if it has been instructed to listen to that address and all frames, if it has been placed in promiscuous mode.

It passes to the host only the frames that it accepts.

Transmitter Algorithm

As we have just seen, the receiver side of the Ethernet protocol is simple; the essence is implemented at the sender's side.

The transmitter algorithm is defined as follows.

When the adaptor has a frame to send and the line is idle, it transmits the frame immediately; there is no negotiation with the other adaptors. The upper bound of 1500 bytes in the message means that the adaptor can occupy the line for only a fixed length of time. When an adaptor has a frame to send and the line is busy, it waits for the line to go idle and then transmits immediately. The Ethernet is said to be a *1-persistent* protocol because an adaptor with a frame to send transmits with probability 1 whenever a busy line goes idle.

In general, a *p-persistent* algorithm transmits with probability $0 \leq p \leq 1$ after a line becomes idle, and defers with probability $q = 1 - p$.

The reasoning behind choosing a $p < 1$ is that there might be multiple adaptors waiting for the busy line to become idle, and we don't want all of them to begin transmitting at the same time.

If each adaptor transmits immediately with a probability of say 33%, then up to three adaptors can be waiting to transmit and the odds are that only one will begin transmitting when the line becomes idle.

Despite this reasoning, an Ethernet adaptor always transmits immediately after noticing that the network has become idle and has been very effective in doing so.

To discuss the entirety of *p*-persistent protocols, we need to see the case when $p < 1$; *i.e.*, how long a sender that decides to defer has to wait before it can transmit.

The solution given by the ALOHA network, which originally developed this style of protocol, was to divide time into discrete slots, with each slot corresponding to the length of time it takes to transmit a full frame. Whenever a node has a frame to send and it senses an empty (idle) slot, it transmits with probability p and defers until the next slot with probability $q = 1 - P$. If that next slot is also empty, the node again decides to transmit or defer, with probabilities p and q, respectively. If that next slot is not empty—that is, some other station has decided to transmit—then the node simply waits for the next idle slot and the algorithm repeats.

Coming back to the Ethernet, because there is no centralized control it is possible for two (or more) adaptors to begin transmitting at the same time, either because both found the line to be idle or because both had been waiting for a busy line to become idle. When this happens, the two (or more) frames are said to *collide* on the network.

Each sender, because the Ethernet supports collision detection, is able to determine that a collision is in progress. At the moment an adaptor detects that its frame is colliding with another, it first makes sure to transmit a 32-bit jamming sequence and then stops the transmission. Thus, a transmitter will minimally send 96 bits in the case of a collision: 64-bit preamble plus 32-bit jamming sequence.

One way that an adaptor will send only 96 bits; which is sometimes called a *runt frame,* is if the two hosts are close to each other. Had the two hosts been farther apart, they would have had to transmit longer, and thus send more bits, before detecting the collision.

In fact, the worst-case scenario happens when the two hosts are at opposite ends of the Ethernet.

To know for sure that the frame it just sent did not collide with another frame, the transmitter may need to send as many as 512 bits. Not coincidentally, every Ethernet frame must be at least 512 bits (64 bytes) long; 14 bytes of header plus 46 bytes of data plus 4 bytes of CRC.

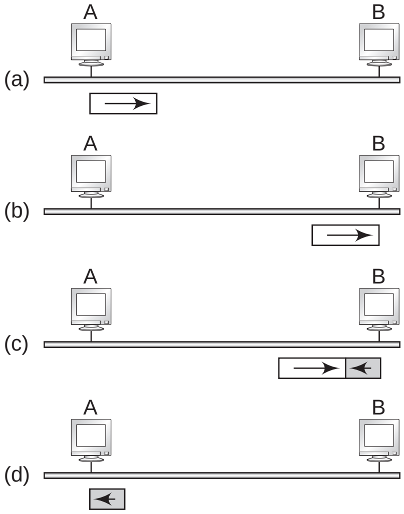


Figure 2.23: Worst-case scenario (a) A sends a frame at time t; (b) A's frame arrives at B at time t + d; (c) B begins transmitting at time t + d and collides with A's frame; (d) B's runt (32-bit) frame arrives at A at time t + 2d

Figure 2.23 illustrates the worst-case scenario, where hosts A and B are at opposite ends of the network.

Suppose host A begins transmitting a frame at time *t*, as shown in (a). It takes it one link latency (let's denote the latency as *d*) for the frame to reach host B. Thus, the first bit of A's frame arrives at B at time *t* +*d*, as shown in (b).

Suppose an instant before host A's frame arrives (*i.e.*, B still sees an idle line), host B begins to transmit its own frame. B's frame will immediately collide with A's frame, and this collision will be detected by host B (c). Host B will send the 32-bit jamming sequence, as described above. (B's frame will be a runt.)

Unfortunately, host A will not know that the collision occurred until B's frame reaches it, which will happen one link-latency later, at time $t + 2 \times d$, as shown in (d). Host A must continue to transmit until this time in order to detect the collision. In other words, host A must transmit for $2 \times d$ to be sure that it detects all possible collisions.

Considering that a maximally configured Ethernet is 2500 m long, and that there may be up to four repeaters between any two hosts, the round-trip delay has been determined to be 51.2 µs, which on a 10-Mbps Ethernet corresponds to 512 bits.

The flip side is that we need to limit the Ethernet's maximum latency to a fairly small value (*e.g.*, 51.2 µs) for the access algorithm to work; hence, an Ethernet's maximum length must be something on the order of 2500 m.

Once an adaptor has detected a collision and stopped its transmission, it waits a certain amount of time and tries again. Each time it tries to transmit but fails, the adaptor doubles the amount of time it waits before trying again. This strategy of doubling the delay interval between each retransmission attempt is a general technique known as ***exponential backoff***.

More precisely, the adaptor first delays either 0 or 51.2 µs, selected at random. If this effort fails, it then waits 0, 51.2, 102.4, or 153.6 µs (selected randomly) before trying again; this is $k \times 51.2$ for $k = 0..3$.

After the third collision, it waits $k \times 51.2$ for k = 0..23 – 1, again selected at random. In general, the algorithm randomly selects a k between 0 and 2n – 1 and waits $k \times 51.2$ µs, where n is the number of collisions experienced so far.

The adaptor gives up after a given number of tries and reports a transmit error to the host. Adaptors typically retry up to 16 times, although the *backoff* algorithm caps n in the above form.

2.6.2 Token Ring

A token ring network is a local area network (LAN) in which all computers are attached in a ring topology. In this method token-passing scheme is used to avoid the collision of data between two computers, which want to send information simultaneously. After Ethernet token ring protocol is the second most widely-used protocol on LAN. Unlike Ethernet, token ring uses a ring topology where the data is sent from one machine to the next. Token passing protocol is used so that a machine can use the network when it has control of the token; this ensures that there are no collisions.

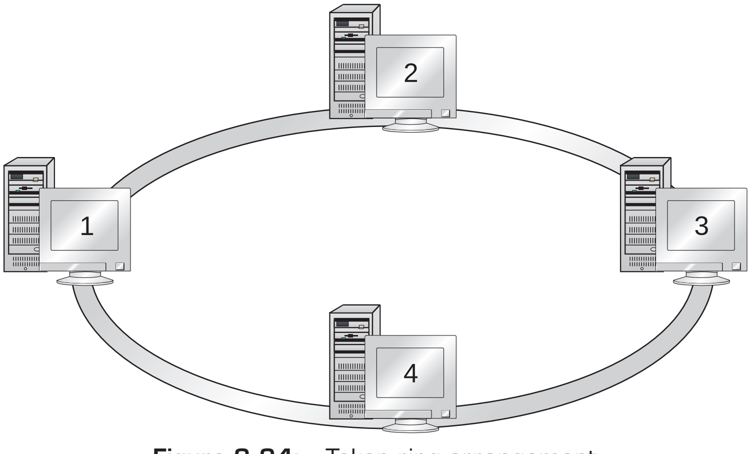


Figure 2.24: Token ring arrangement

Physical Properties

One of the first things you might worry about with a ring topology is that any link or node failure would render the whole network useless. This problem is addressed by connecting each station into the ring using an electromechanical relay. As long as the station is healthy, the relay is open and the station is included in the ring.

If the station stops providing power, the relay closes and the ring automatically bypasses the station. This is illustrated in Figure 2.25.

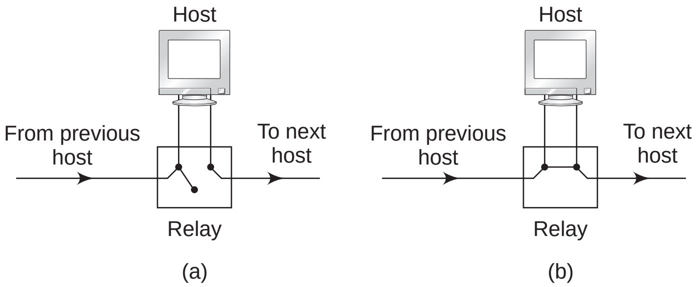


Figure 2.25: Relay used on a token ring: (a) relay open—host active; (b) relay closed—host bypassed

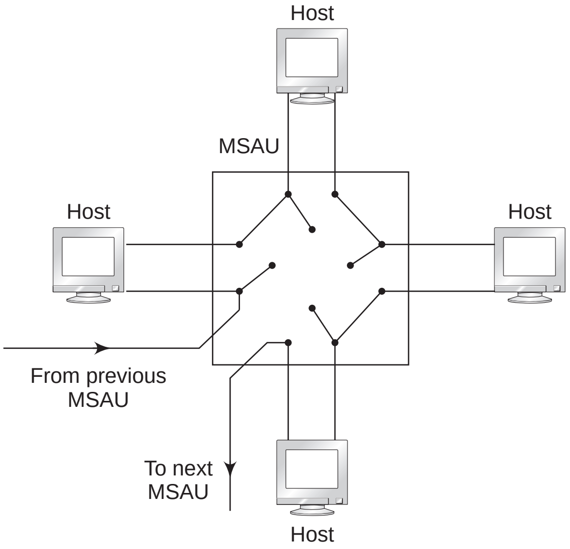


Figure 2.26: Multistation access unit

Several of these relays are usually packed into a single box, known as a **multi-station access unit** (MSAU). This has the interesting effect of making a token ring actually look more like a star topology, as shown in Figure 2.26. It also makes it very easy to add stations to and remove stations from the network, since they can just be plugged into or unplugged from the nearest MSAU, while the overall wiring of the network can be left unchanged.

Token Ring Media Access Control

Let us now look a little more closely at how the MAC protocol operates on a token ring.

The network adaptor for a token ring contains a receiver, a transmitter, and one or more bits of data storage between them. When none of the stations connected to the ring has anything to send, the token circulates around the ring. Obviously, the ring has to have enough "storage capacity" to hold an entire token.

For example, the 802.5 token is 24 bits long. If every station could hold only 1 bit (as is the norm for 802.5 networks), and the stations were close enough together that the time for a bit to propagate from one station to another was negligible, we would need to have at least 24 stations on the ring before it would operate correctly. This situation is avoided by having one designated station, called the monitor add some additional bits of delay to the ring if necessary.

As the token circulates around the ring, any station that has data to send may “grab” the token, that is, drain it off the ring and begin sending data. In 802.5 networks, the *grabbing* process involves simply modifying 1 bit in the second byte token; the first 2 bytes of the modified token now become the preamble for the subsequent data packet.

Once a station has the token, it is allowed to send one or more packets—exactly how many more depends on some factors described below. Each transmitted packet contains the destination address of the intended receiver; it may also contain a multicast (or broadcast) address if it is intended to reach more than one (or all) receivers.

As the packet flows past each node on the ring, each node looks inside the packet to see if it is the intended recipient. If so, it copies the packet into a buffer as it flows through the network adaptor, but it does not remove the packet from the ring.

The sending station has the responsibility of removing the packet from the ring. For any packet that is longer than the number of bits that can be stored in the ring, the sending station will be draining the first part of the packet from the ring while still transmitting the latter part.

One issue we must address is how much data a given node is allowed to transmit each time it possesses the token, or said another way, how long a given node is allowed to hold the token.

We call this the ***token holding time*** (THT). If we assume that most nodes on the network do not have data to send at any given time—a reasonable assumption, and certainly one that the Ethernet takes advantage of—then we could make a case for letting a node that possesses the token transmit as much data as it has before passing the token on to the next node. This would mean setting the THT to infinity.

It would be silly in this case to limit a node to sending a single message and to force it to wait until the token circulates all the way around the ring before getting a chance to send another message. Of course, “as much data as it has” would be dangerous because a single station could keep the token for an arbitrarily long time, but we could certainly set the THT to significantly more than the time to send one packet.

It is easy to see that the more bytes a node can send each time it has the token, the better the utilization of the ring you can achieve in the situation in which only a single node has data to send. The downside, of course, is that this strategy does not work well when multiple nodes have data to send—it favors nodes that have a lot of data to send over nodes that have only a small message to send, even when it is important.

2.6.3 FDDI (Fiber Distributed Data Interface)

Fiber Distributed Data Interface (FDDI) is a set of ANSI protocols for sending digital data over fiber optic cable. FDDI supports data rates of up to 100 Mbps. FDDI networks are considered to very important because of its support for high bandwidth and great distance. There is a related copper specification available, called Copper Distributed Data Interface (CDDI). It is similar to FDDI protocols and it has also been defined to provide 100-Mbps service over twisted-pair copper. FDDI-2 is the successor of FDDI. FDDI-2 supports the transmission of voice and video information as well as data. Another variation of FDDI called FDDI Full Duplex Technology (FFDT) uses the same network infrastructure but can potentially support data rates up to 200 Mbps.

FDDI uses dual-ring architecture, where both ring travel in opposite directions. The dual rings consist of a primary and a secondary ring. During normal operation, the primary ring is used for data transmission, and the secondary ring remains idle. The primary purpose of the dual rings is to provide superior reliability and strength.

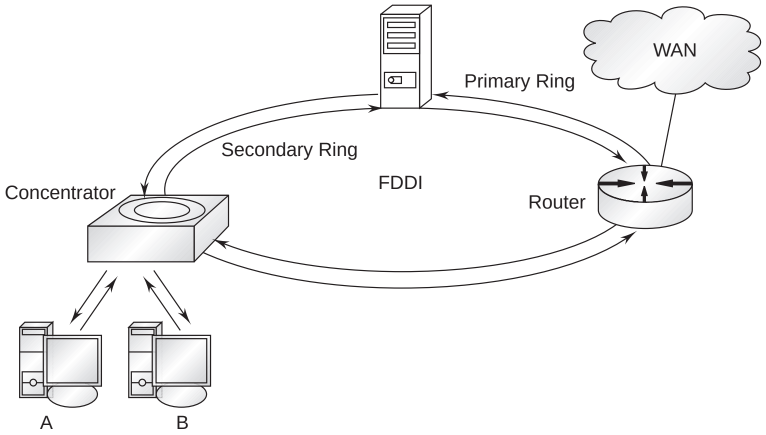


Figure 2.27: FDDI architecture

FDDI identifies two types of optical fiber: single-mode and multimode. A mode is a ray of light that enters the fiber at a particular angle. *Multimode* fiber uses LED as the light-generating device, while *single-mode* fiber generally uses lasers. Multimode fiber allows multiple modes of light to broadcast through the fiber. Single-mode fiber allows only one mode of light to broadcast through the fiber. Figure below depicts single-mode fiber using a laser light source and multimode fiber using a light emitting diode (LED) light source.

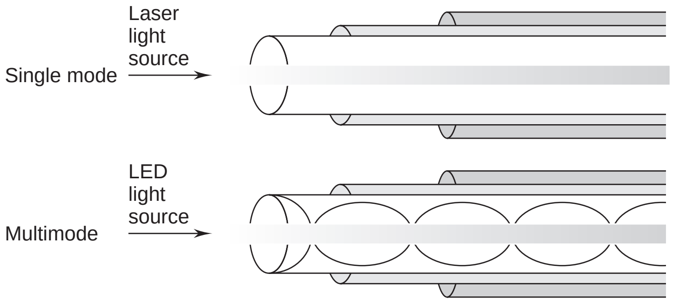


Figure 2.28: Fiber media

There are four specifications in FDDI. They are:

- **Media Access Control (MAC) –** The *MAC* specification defines how the medium is accessed, including frame format, token handling, addressing, algorithms for calculating cyclic redundancy check (CRC) value, and error-recovery mechanisms.
- **Physical Layer Protocol (PHY) –** The*PHY* specification defines data encoding/decoding procedures, clocking requirements, and framing, among other functions.
- **Physical-Medium Dependent (PMD) –** The *PMD* specification defines the characteristics of the transmission medium, including fiber-optic links, power levels, bit-error rates, optical components, and connectors.

- **Station Management (SMT)** – The *SMT* specification defines FDDI station configuration, ring configuration, and ring control features, including station insertion and removal, initialization, fault isolation and recovery, scheduling, and statistics collection.

FDDI data frames are similar to IEEE 802.3 Ethernet and IEEE 802.5 Token Ring in its relationship with the OSI model. Its primary purpose is to provide connectivity between upper OSI layers of common protocols and the media used to connect network devices. Figure below shows the four FDDI specifications and their relationship to each other and to the IEEE-defined Logical Link Control (LLC) sub-layer.

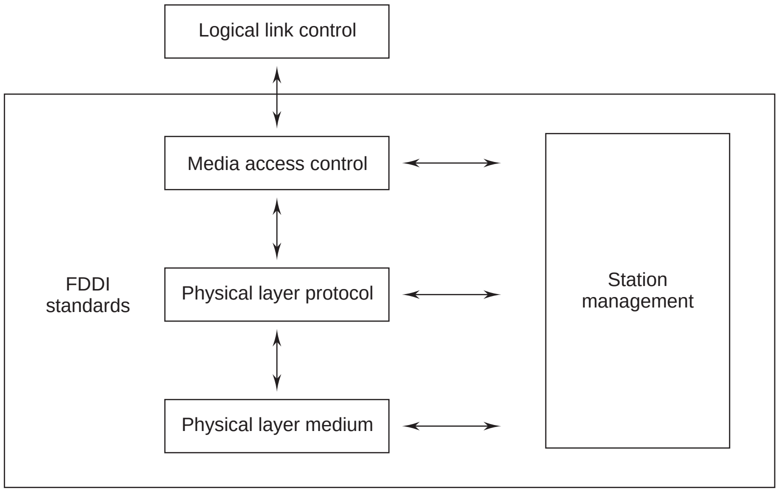


Figure 2.29: FDDI specifications

2.7 WIRELESS LANS

Wireless communication is one of the fastest growing technologies. The need for connecting devices without the use of cables is increasing everywhere. Wireless LAN is an alternative to the network cables used to connect computers at home and office. A wireless LAN should meet the same kind of requirements of any LAN, which includes high capacity, ability to cover short distances, full connectivity among attached stations and broadcast capability. Wireless LAN data transfer speeds vary from 1 Mbps to 54 Mbps.

Wireless LAN may need to support hundreds of nodes across multiple cells. To maximize the capacity the medium access control protocol should make as much use as possible of the wireless medium. A coverage area for a wireless LAN is diameter of 100 to 300 meters. The design of a wireless LAN must allow reliable transmission even in a noisy environment and it should provide some level of security from eavesdropping.

Wireless LANs are generally grouped according to the transmission technique that is used. They are,

- **Infrared (IR) LANs** – IR LAN is limited to a single room, because infrared light does not penetrate opaque walls.
- **Spread spectrum LANs** – this type of LAN makes use of spread spectrum transmission technology.
- **Narrowband microwave** – these LANs operate at microwave frequencies but do not use spread spectrum.

Standards for Wireless LAN

IEEE has defined the specifications for a wireless LAN, called IEEE 802.11, which covers the physical and data link layer. IEEE 802.11 developed a set of wireless LAN standards. The standard defines two kinds of services. They are the basic service set (BSS) and the extended services set (ESS).

- IEEE 802.11 defines the BSS as the building block of a wireless LAN. A basic service set is made of stationary or mobile wireless stations and an optional central base station, known as the access point (AP).

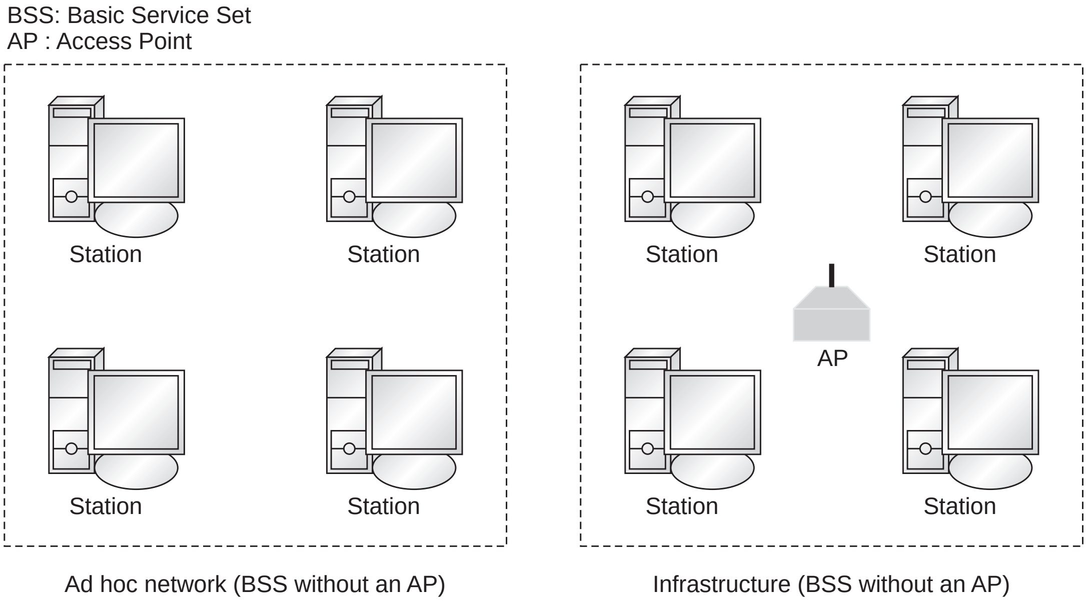


Figure 2.30: Building blocks of wireless LAN

- An ESS is made up of two or more BSSs with APs. In this method, the BSSs are connected through a distribution system, which is usually a wired LAN. The distribution system connects the APs in the BSSs.

The standard we choose is depends on our needs. The concern lies on speed or compatibility or price. In the following section we look into four major wireless networking standards.

802.11a – It is one of several specifications in the 802.11 group relevant to wireless local area networks. Networks using 802.11a operate at radio frequencies between 5.725 GHz and 5.850 GHz. The specification uses a modulation scheme known as orthogonal frequency-division multiplexing (OFDM) that is especially well suited to use in office settings. In 802.11a, data speeds as high as 54 Mbps are possible. Compare with 802.11b, there is less interference with 802.11a because 802.11a provides more available channels.Hence it offers bigger bandwidth and fewer interference problems but a shorter range.

802.11b – This standard is the corporate friendly and has a suitably wide range for use in big office spaces. It is often called Wi-Fi. It is part of the 802.11 series of WLAN standards from IEEE. It is backward compatible with 802.11. 802.11b uses the Ethernet protocol and CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) for path sharing.

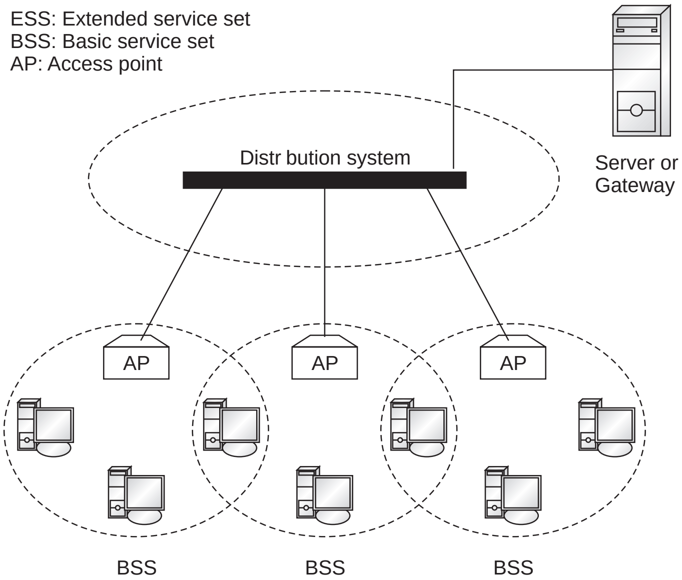


Figure 2.31: Wireless LAN architecture

802.11g — a new upcoming standard, an extension of the 802.11b standard, which means that old 802.11b equipment, will work with the new 802.11g equipment. It offers transmission over relatively short distances at up to 54 megabits per second (Mbps). Network employing 802.11g operate at radio frequencies between 2.400 GHz and 2.4835 GHz. The 802.11g specification employs orthogonal frequency division multiplexing (OFDM), the modulation used in 802.11a to get higher data speed.

Bluetooth — is meant for short-range, temporary networking in conference rooms, schools, or homes. Bluetooth is an industrial specification for wireless personal area networks (PANs). It is used to connect and exchange information between devices. For example; devices such as mobile phones, laptops, PCs, printers, digital cameras and video game consoles. This technology works through globally unlicensed short range radio frequency.

2.7.1 Topology of Wireless LAN

Wireless LAN can be used in two ways. One is *peer-to-peer* or using a central *Access Point*.

Peer-to-Peer – It is the simplest way to connect PCs using wireless. Sometimes it called Point-to-Point. It allows wireless devices to directly communicate with each other. Wireless devices within particular range of each other can iscover and communicate directly without involving central access points. This method is typically used by two computers and only one need to buy cards for the PC. There is no need for a hub or switch if one PC connected to another.

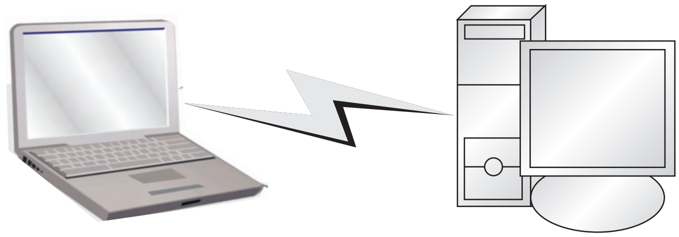


Figure 2.32(a): Peer-to-peer wireless LAN

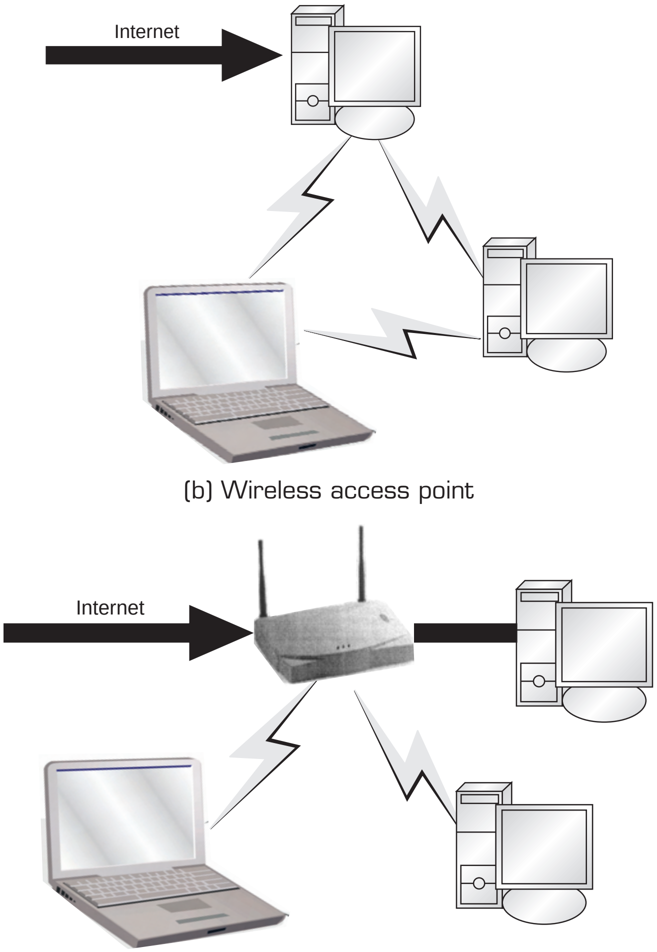
As shown in the figure below if three computers are connected and added an Internet connection, the Internet is connected to one PC. That particular computer has software running to share this connection.

For example: Microsoft's Internet Connection Sharing.

Access Point – Wireless access points are specially configured nodes on wireless local area network (WLAN). Access point acts as a central transmitter and receiver of WLAN radio signals.

Access point (AP) is bit more complicated. Here we use a central box (AP) that handles traffic. These boxes act as a kind of bridge between the connected PCs. The AP is not just a hub, but more like a router that has functionality like NAT that is sharing one IP address over multiple PCs, DHCP that is managing the available IP range, Firewall, etc. The advantage of AP is that an AP directs its radio signals in multiple directions and has a better range. Some APs even supports additional external antennas for better range.

Though the range is high, the fact is that the available bandwidth has to be shared with other users. The figure below shows that the AP handles Internet traffic as well, and can handle fixed connections too (the straight line).



(b) Wireless access point

(c) Hybrid access

Figure 2.32: Wireless LAN

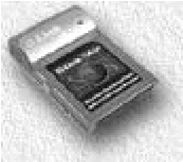
2.7.2 Available Equipment

There is lot of equipment that can be used to connect to Wireless LAN.

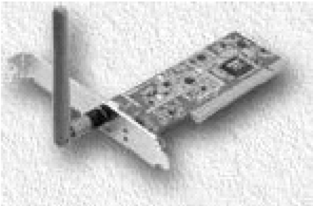
Laptop – Today more laptops are come with a built in Wi-Fi facility. This can be part of the chipset or this can be mini-PCI card. The advantage is that there are no parts sticking out of the laptop and generally better antenna than a PCMCIA card.

PCMCIA Cards – This is commonly used with laptops. Sometimes it is used in combination with a PCI adaptor or as an add-on for some routers. PCMCIA cards are inserted in a slot on the side of the laptop. Some cards have a little connector to add an additional external antenna for a better reception.

Compact Flash Cards – These cards are commonly used with PDAs (Personal Digital Assistants). PDA usually has a slot on the top where we can insert compact flash card.

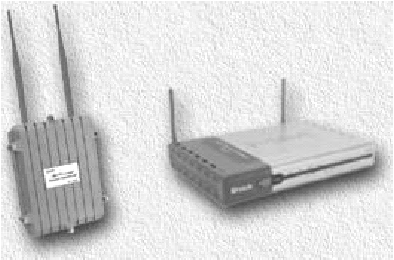


PCI Card – This card is used only for desktops. This comes in two shapes. One is as an adaptor to carry a PCMCIA or as a native Wi-Fi PCI card.



USB– This device we can directly inserted into the USB port or as a separate device connected with a cable to the USB port. USB devices are commonly used with desktops and laptops. Some PDA can handle them as well. Most of the USB devices take power from the USB port. Some USB devices need an additional power supply for powering.

AP: Access Point– AP comes in different shapes and size. Besides the normal APs, there are also special outdoor APs.



Camera – There are other devices apart from a PC that can be connected to the Wi-Fi network, for example a camera.

- The data link layer is designed to give various services but the services given by data link layer differ from one system to another.
- Framing in the data link layer differentiate one message from a particular source to a destination or from some other message to other destinations.

- The data carried in a character-oriented protocol are 8-bit character from a coding system such as ASCII.
- In bit-oriented protocols the data section of a frame is a series of bits to be interpreted by the upper layer as text, graphic, audio, video, and other data.
- The most important tasks of the data link layer are *error control* and *flow control*
- Flow control controls the amount of data that can be sent before receiving an acknowledgment.
- Logical Link Control is the upper sub-layer of the OSI data link layer.
- Media Access Control is the sub-layer of data link layer.
- The main concept of detecting or correcting error is based on *redundancy.*
- By using different coding schemes, redundancy is achieved
- IEEE has produced many standards for LANs. These standards together called as IEEE 802.
- The CSMA (Carrier Sense Multiple Access) method is developed to reduce the chance of collision hence increase the performance.
- The difference between bus and ring is the bus does not meet to form a physical ring.
- A token ring network is a local area network (LAN) in which all computers are attached in a ring topology.

3

INTERNETWORKING

Introduction

Networks connect devices and computers to one other thus enabling the sharing of resources. Connecting these multiple devices so that there is one-to-one communication between them is always a hassle. Many topologies and designs exist by which this communication is made possible. There is the mesh topology wherein devices are connected point-to-point with each device being connected to every other device on the network.

There is also the star topology whereby every device on the network is connected to a central device, thus making the network look star-like. These methods have their advantages but seem impractical in large networks. The bus topology is impractical also as the number of devices and the distance between them will be so huge that the equipment and media used will not be able to support it.

When it comes to large networks, having links for a point-to-point connection would need a lot of infrastructure. Taking the number and length of links that would be required in a large network into consideration; you will realize that the network is not cost-efficient. Additionally, most of these links will not be used all the time.

This is where ***switching*** comes into the picture.

3.1 SWITCHING

Switches basically change the direction of flow of data by making temporary connections between devices that have been connected to the switch. These interlinked nodes thus reduce the need to have as many links as required for a point-to-point connected network. In a switched network, you can see that some of the switches are connected to end systems like the computer or a telephone and others are used for the sole function of routing.

The following figure shows one such switched network where the end systems are labeled as A,B,C,D etc while the switches are labeled as 1,2,3 etc.

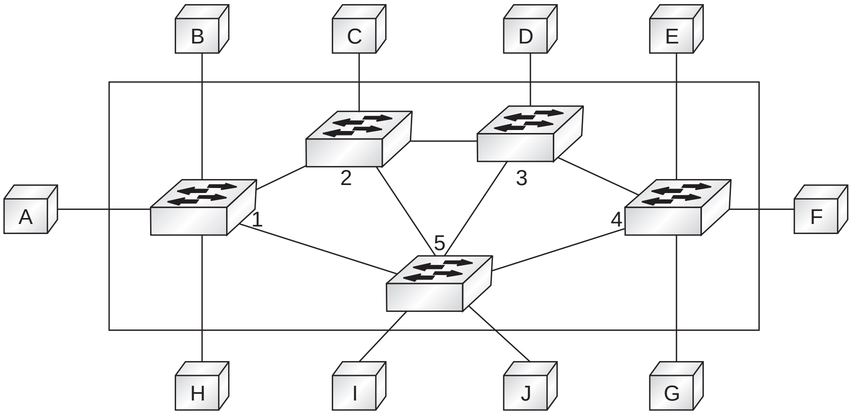


Figure 3.1: Switched network

Switching can be broadly classified into three categories:

- Circuit Switching
- Packet Switching
- Message Switching

Of the three, message switching is no longer in use in the field of general communications. It is still relevant for networking applications though. Packet switching can be further divided into:

- Datagram Networks
- Virtual Circuit networks

The following figure explains the taxonomy of switched networks:

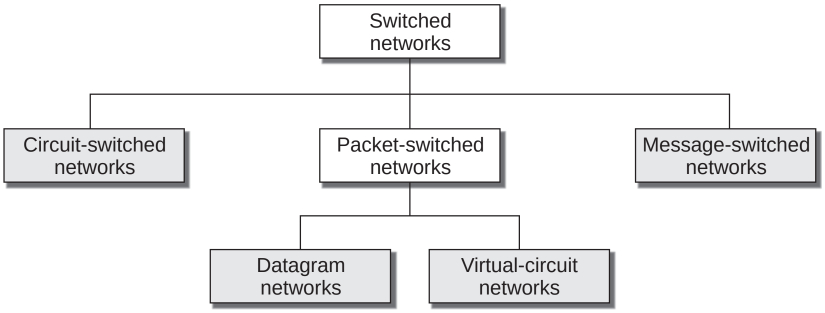


Figure 3.2: Taxonomy of switched networks

The sub-categorization in packet switching is not followed these days; instead a combination of both datagram networks and virtual circuit technology is used. This is done in the following way:

The first packet is routed using the addressing method used for datagram networks. A virtual circuit is then created for the remaining packets to be sent to the same destination from the same source.

Message switching actually came before packet switching and is also known as ***Store and forward*** switching. No physical path is created in advance between the sender and receiver in this process. The entire block of data to be sent is *stored* in the switching office (or the router) and then *forwarded* later to the next switch.

Message switching is used only in some applications like electronic mail these days and not in the lower layers.

3.1.1 Circuit switching and Packet Switching

Table 3.1: Different switching mechanisms

Circuit Switching	Packet Switching
Initial delay when setting up a connection	Minimal set up delay
Reliable connection	Congestion chances higher thus leading to dropping of packets
Dedicated resources; therefore inefficient usage	Efficient shared use of resources

3.1.2 Datagram Networks

Packet switching refers to protocols in which messages are broken up into small ***packets*** before they are sent. Each packet is transmitted individually across the net, and may even follow different routes to the destination. At the receiving station, the reverse happens-the packets are reassembled into their original format Thus, each packet has header information about the source, destination, packet numbering, etc. At the destination the packets are reassembled into the original message.

Most modern Wide Area Networks (WANs) protocols, such as TCP/IP, X.25 and Frame Relay, are based on packet switching technologies. The following figure shows how blocks of data can take any path to reach their destination.

The main difference Packet switching has to Circuit Switching is that the communication lines are not dedicated to passing messages from the source to the destination. In Packet Switching, different messages (and even different packets) can pass through different routes, and when there is a "dead time" in the communication between the source and the destination, the lines can be used by other routers.

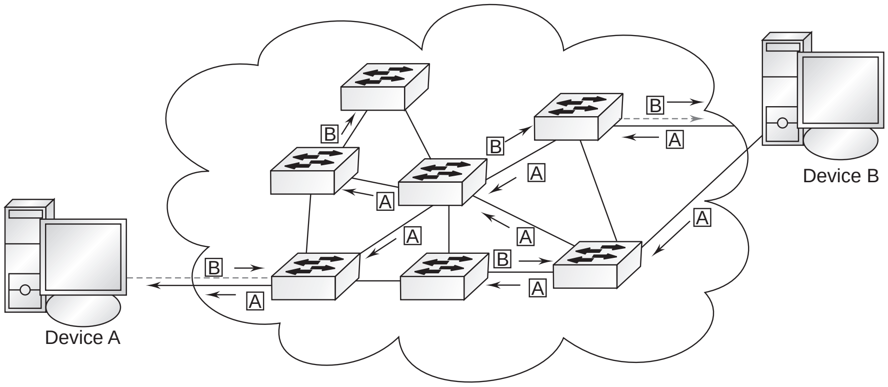


Figure 3.3: Datagram switching

Datagram switching, a sub-division of packet switching, is done at the *network layer.* As you know, data is chopped up into packets before being sent over the packet-switched network. Each of these packets, even if it is just one piece of a huge multi-packet transmission, is considered as a separate entity in a datagram network. These packets are hereby referred to as ***datagrams.***

Datagram networks are also known as ***connectionless networks*** as the switches do not retain information on the connection state. There are also no setup or teardown phases, which were existent in circuit switched networks.

The following figure shows a typical datagram network.Four packets are being sent from System A to System B. The switches are known as routers in a datagram network.

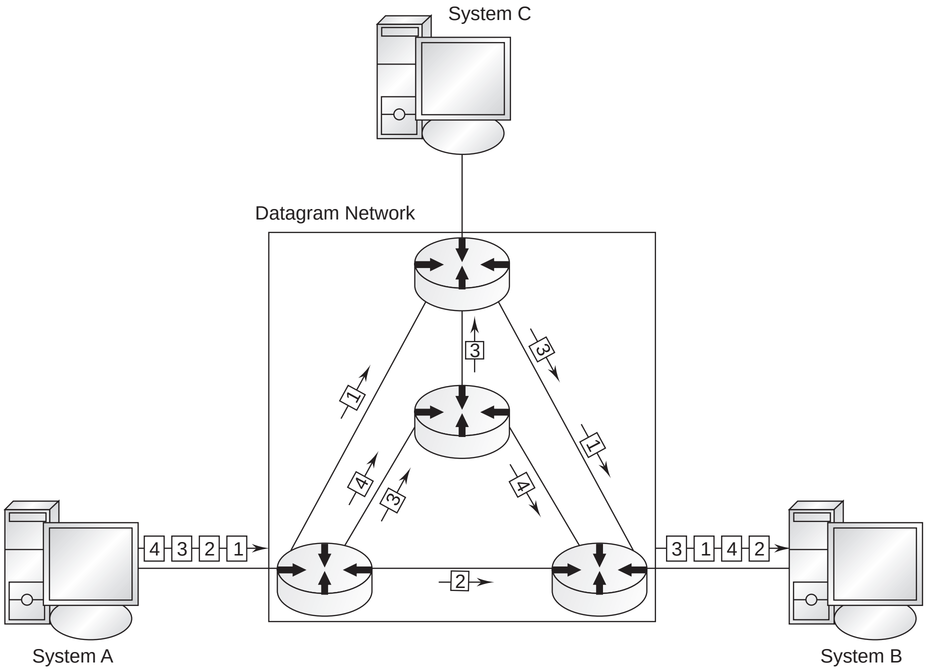


Figure 3.4: Datagram network

The data has been chopped up into four packets and they can take various paths to reach System B. This primarily occurs because some links may not have the required bandwidth to carry the packets as they may be carrying packets from some other source as well. This causes the packets to reach the destination station in a mixed up manner. It is also possible that some of the packets are lost or even dropped during the transmission. The protocol working at the upper layers does the task of re-organizing the packets or even requesting for missing packets, if any.

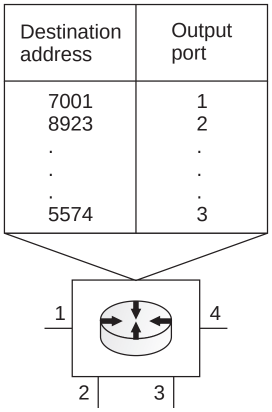
3.1.2.1 Routing Table

The concept of datagram networks can be visualized using the simple example of the postal service. When people want to send mail, they do not have to create a connection beforehand. They just write the address of the desired receiver and mail their letter at a post office closest to them. The post office then forwards the mail to another post office which would be located closer to the destination. In this way, the mail goes from one post office to another till it finally reaches the local post office of the addressee.

The datagram network works in the same way: the sender transmits datagrams that include the address of the destination in the header; datagrams also usually include the sender's address to help the receiver send messages back to the sender. The job of the switch is to use the destination address as a key and perform a lookup on a data structure called a ***routing table***. This lookup returns an output port to forward the packet on towards the intended destination.

Datagram switching does not involve the setup or tear down phases which were seen in circuit switched networks. The routing tables that enable the routing of datagrams are dynamic in nature and are updated at regular intervals of time. These tables have a record of destination addresses and their corresponding forwarding output ports.

A routing table in circuit switching, unlike datagram switching, has its entries created at the end of the setup phase but deleted after the teardown phase. A routing table is shown in the following table.



Destination address	Output port
7001	1
8923	2
.	.
.	.
.	.
5574	3

Figure 3.5: Routing table in circuit switching

The destination address seen in the header of a packet remains the same throughout till the packet reaches its destination.

3.1.2.2 Efficiency

Since resources like bandwidth, channels etc are allocated only on demand, the efficiency of a datagram network is far better than that of a circuit switched network. Take the case where the sending station could send the first and second packet but experienced a bit of delay when sending the third packet. The resources get reallocated and are used to send other packets in the mean time thus ensuring that none of the resources are left idle and unused.

3.1.2.3 Delay

Even though there are no setup or teardown phases in datagram switching, there is a considerable amount of delay which can be greater than the delay seen in a virtual-circuit network. The delay is due to the wait time that each packet will face at every switch it encounters in its journey towards the destination. This wait time will vary as packets may not necessarily travel through the same switches. Thus the delay will not be the same for every packet in a message.

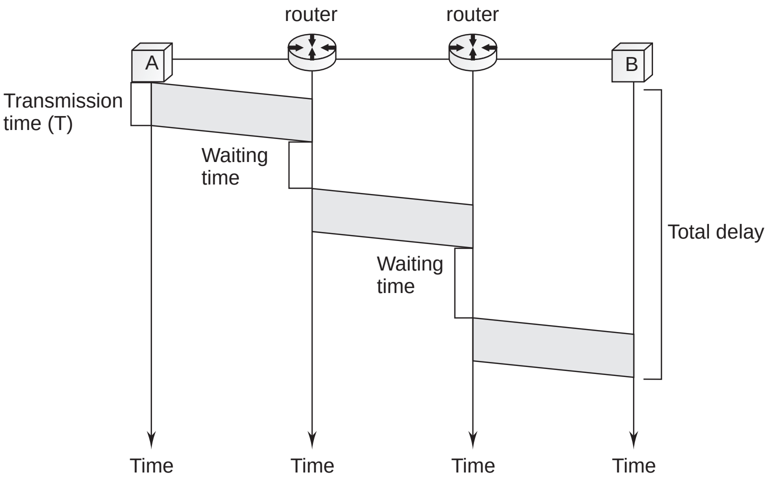


Figure 3.6: Delay in a virtual circuit

The above graph shows the delay for a single packet in a datagram network. The network has just two switches but note the waiting time (which can vary) at each switch. The total delay would be the sum of the total transmission time (height of the boxes), total propagation delays (slope of the boxes) and the total waiting time at the two switches.

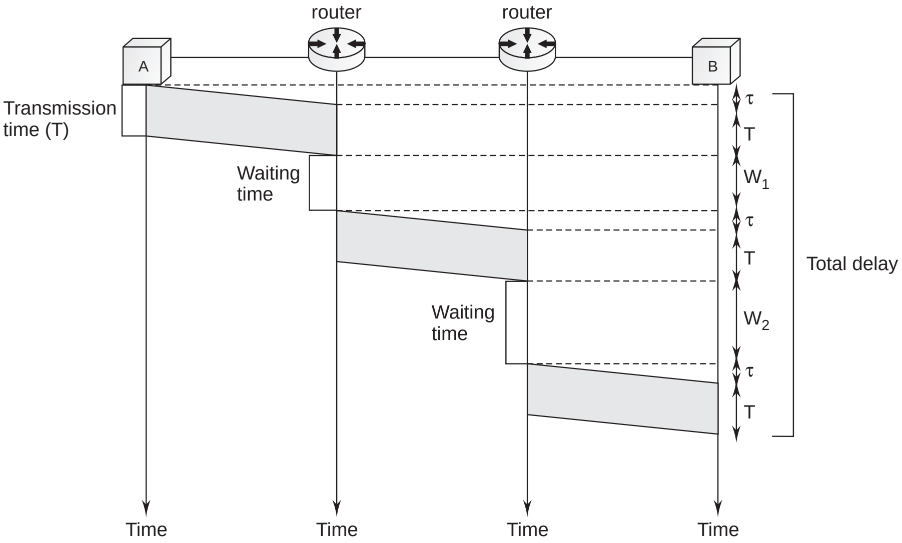
If the height of each box is taken as **T**, then **total transmission time = 3T**

Slope of each box is taken as τ, then **total propagation delay = 3 τ**

Waiting time at switch 1 is taken as $\mathbf{w_1}$

Waiting time at switch 2 is taken as $\mathbf{w_2}$

$$\boxed{\textbf{Total delay} = 3T + 3\tau + w_1 + w_2}$$



Total Delay = $3T + 3\tau + W_1 + W_2$

Figure 3.7: Calculation of total delay

3.1.3 Virtual-Circuit Networks

Virtual circuit packet switching (VC-switching) is a packet switching technique which merges datagram packet switching and circuit switching to extract both of their advantages. The set up and teardown phases exist in a virtual circuit network, just like in the circuit switched network.

The sending station tries to create a *virtual circuit* with the receiver. If this attempt is successful, then a different ***virtual circuit identifier (VCI)*** is allocated to each of the network devices taking part in the communication. It is this virtual circuit number that becomes the packet's address as it traverses the network. So, just like the datagram network, the packets have an address in its header but, this address (which includes the VCI) decides which switch and channel it should be forwarded on to and does not have end to end addressing.

Allocation of resources can occur at the time of the setup phase (just like in the circuit switched network) or on demand (like the datagram network). The path that was created during the connection is followed by all packets, just like in circuit switched networks. Implementation of virtual circuit networks is at the *data link layer* when compared to the circuit switched network (physical layer) or the datagram network (network layer).

The most common form of virtual circuit networks were ATM and X.25.

3.1.3.1 Addressing

Virtual Circuit networks require two types of addresses:

- **Global Address:** If the network is to be part of a bigger or an international network, the sending station or the destination station will require a global address that is unique internationally. If the network is not part of an international network, the global address should be unique to the scope of the network. The global address is mainly used to create the virtual circuit identifier (VCI).
- **Local address or Virtual Circuit Identifier (VCI):** The Virtual Circuit Identifier is the *real* address that is important for data transfer. A VCI is only used in between two switches. As shown in the following figure, when a frame with datareaches a switch, it has one VCI and when it leaves, it has a new VCI. The VCI does not have to be a huge number as it is unique between switches.

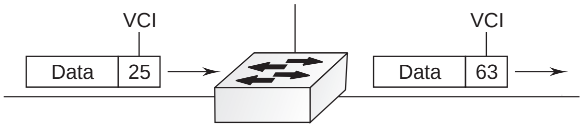


Figure 3.8: Addressing using VCI

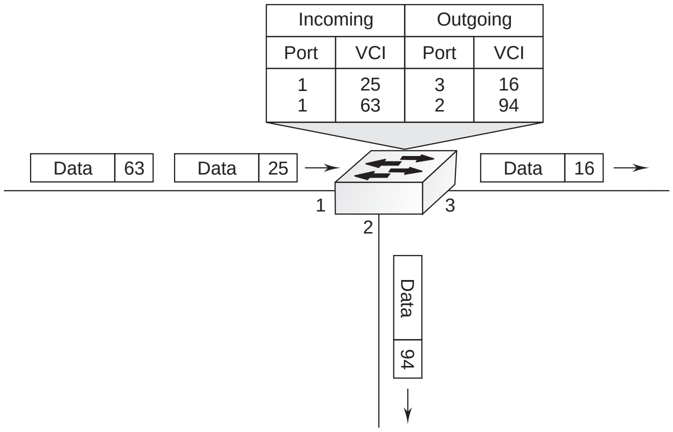
3.1.3.2 Communication Phases

In order for stations to communicate with each other, three phases need to take place. Just like in the circuit switched networks, the three phases are:

- **Setup Phase:** In the duration of this phase, the sending and receiving stations use their global addresses to enable switches to make entries in their switching tables.
- **Data Transfer Phase**
- **Teardown Phase:** At this phase, the sending station and the receiving station lets the switches know that they can erase the specific entries.

Let us discuss each of these phases in detail. We will also go into the exact process that occurs in them. Since the data transfer phase is the simplest, let us explain it first.

1. **Data Transfer Phase:** Every switch in the proposed virtual circuit should have the table entries specific to the circuit ready if a frame has to be transmitted. Consider the table seen in the following diagram. It has basically four columns: the incoming frame's VCI, the incoming frame port, the outgoing frame's VCI, and the outgoing frame port.



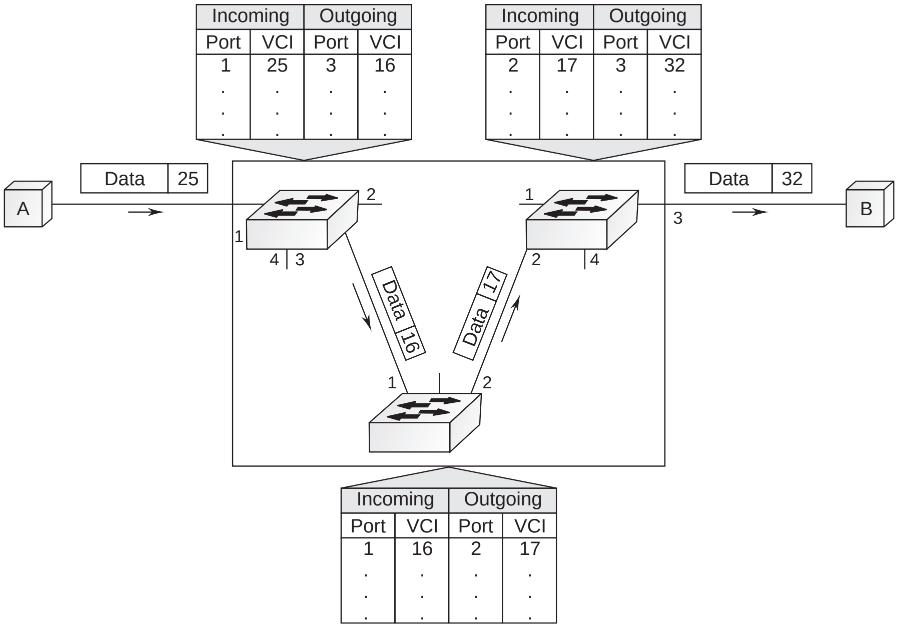
Incoming		Outgoing	
Port	VCI	Port	VCI
1	25	3	16
1	63	2	94

Figure 3.9: Routing table during data transfer

These entries are filled up during the setup phase. When a frame with a VCI of 25 reaches the switch on port 1, the switch checks the contents of its table and looks for an entry for port 1 and a VCI of 25. On discovering the entry, it realizes that it should forward the frame onto port 3 with a new VCI of 16.

The next figure shows the entire process of a frame traveling from the sender A to the destination B. Note how the VCI of the frame changes each time it passes a switch on the way.

This phase continues until all the frames to be sent are transferred over to the destination.



Incoming		Outgoing	
Port	VCI	Port	VCI
1	25	3	16
.	.	.	.
.	.	.	.
.	.	.	.

Incoming		Outgoing	
Port	VCI	Port	VCI
2	17	3	32
.	.	.	.
.	.	.	.
.	.	.	.

Incoming		Outgoing	
Port	VCI	Port	VCI
1	16	2	17
.	.	.	.
.	.	.	.
.	.	.	.

Figure 3.10(a): Data transfer using VCI

2. **Setup Phase:** When creating a *virtual* circuit between the sender and the receiver, the switches participating in the virtual circuit need to have the required circuit entries. The writing of these entries occurs in the setup phase.

There are two steps in the setup phase: Setup request and Acknowledgement.

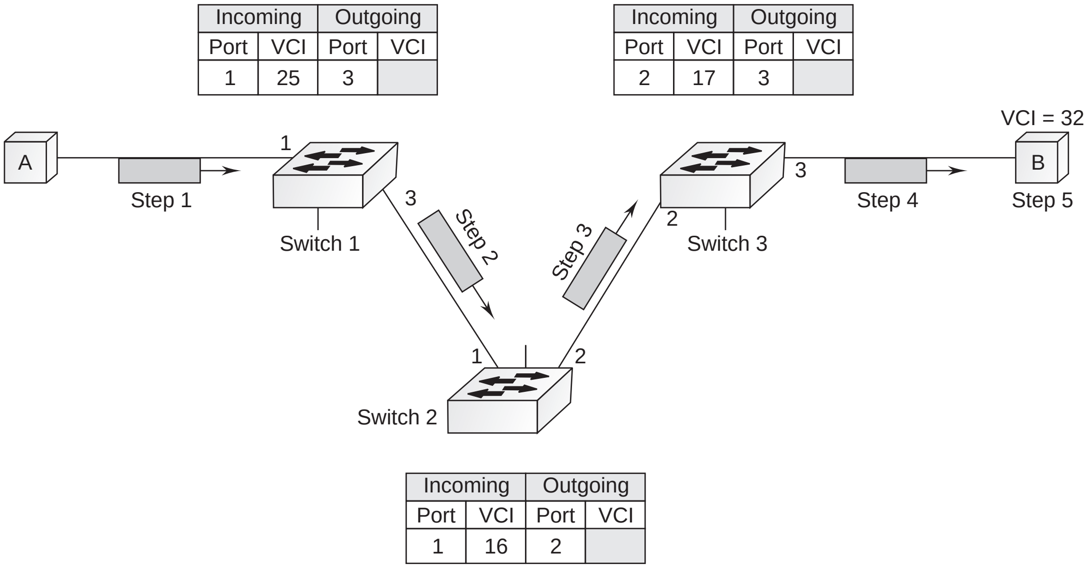
- **Setup Request:** The sending station A sends a setup request frame to the receiver B. The following points occur in succession in this phase.

  Step 1: The setup frame first goes to Switch 1. It knows that any frame going from A to B must be put out on port 3. The switch can only fill up three of the four columns as it still does not know the outgoing VCI as yet. (This will be found out during the acknowledgment phase). Thus the only entries it can make at present are: IncomingPort =1, Incoming frame's VCI =25 and outgoing port=3. It thus sends the frame out on port 3 towards switch 2.

  Step 2: Switch 2 receives the frame and the same sequence of events occur here. Only three out of four entries are filled up. Incoming port =1, incoming VCI=16, and outgoing port =2.

  Step 3: The same procedure occurs at switch 3. It enters the incoming and outgoing port to be 2 and 3 respectively. The incoming VCI is 17.

  Step 4: Once B receives the frame, and if it can receive frames from the sender A, it first allocates a VCI to the frames that come from A. In this example, the allocated VCI for frames from A is 32. By doing this, the receiver knows that frames with VCI=32 is from A and nobody else.



Switch 1:

Incoming		Outgoing	
Port	VCI	Port	VCI
1	25	3	

Switch 3:

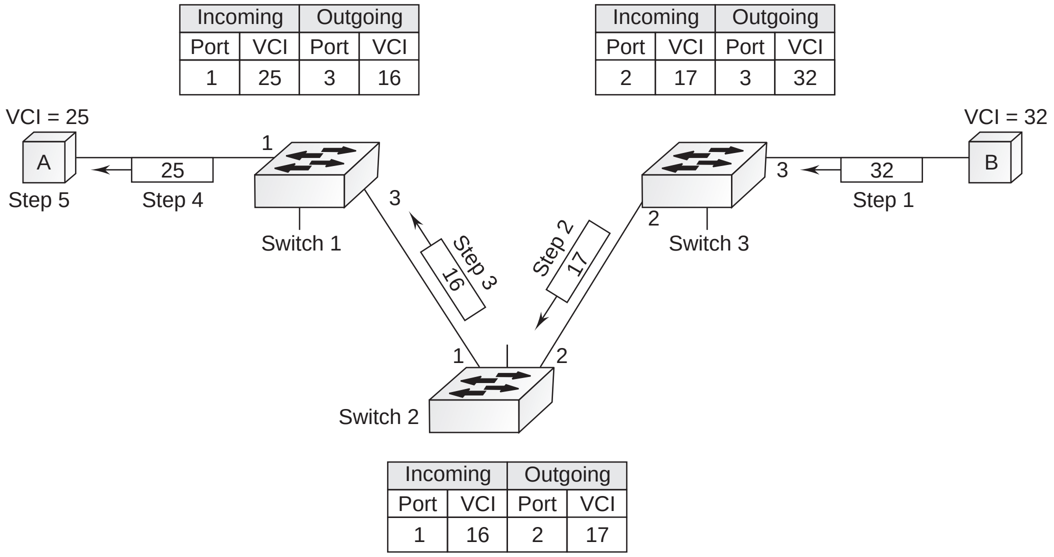
Incoming		Outgoing	
Port	VCI	Port	VCI
2	17	3	

Switch 2:

Incoming		Outgoing	
Port	VCI	Port	VCI
1	16	2	

Figure 3.10(b): Setup request using VCI

- **Acknowledgement:** The incomplete entries in the switching tables are now filled in using the acknowledgement frame.



Switch 1:

Incoming		Outgoing	
Port	VCI	Port	VCI
1	25	3	16

Switch 3:

Incoming		Outgoing	
Port	VCI	Port	VCI
2	17	3	32

Switch 2:

Incoming		Outgoing	
Port	VCI	Port	VCI
1	16	2	17

Figure 3.10(c): Acknowledgement using VCI

Step 1: The acknowledgement frame that is sent back from B to switch 3 will contain 3 things:

- o Source's global address
- o Destination's global address
- o VCI selected by B for all frames from A (VCI=32)

The switch knows which entry needs to be filled in when it views the global addresses. The VCI is used to fill in the remaining column (outgoing VCI) in its switching table. Thus, the outgoing VCI for switch 3 (32) is the incoming VCI for receiver B.

Step 2: Now, switch 3 sends an acknowledgment frame to switch 2 with its incoming VCI (=17). Switch 2 thus uses this VCI to fill up its outgoing VCI column.

Step 3: Similarly, switch 2 sends an acknowledgment to switch 1 with its incoming VCI (=16). Switch 1 uses it to complete its switching table.

Step 4: Switch 1 now sends an acknowledgement to the sender A with its incoming VCI (=25).

Step 5: Sender A will use this VCI=25 as its outgoing VCI for data frames to be sent to B.

3. **Teardown Phase:** The sender A will send out a request known as the *teardown request* when the transmission of data frames to the receiver is complete. The receiver will reply with a teardown confirmation frame, after which, all switches in the virtual circuit proceed to delete the entries specific to that circuit from their switching tables.

3.1.3.3 Efficiency of a Virtual Circuit Network

Allocation of resources in a virtual circuit network can be done in two ways:

- During the setup phase: This causes each packet to have the same delay. This causes the packets from the same sender to the same receiver to traverse the same path.

or

- On demand during the data transfer phase: Different delays will be seen here for the various packets thus causing the packets to arrive at the receiver's end at different delay timings.

Therefore the delay for the second option will be larger than if the resources were allocated in advance. But, there is a benefit that can be seen if the resources are allocated on demand. It saves on a lot of time as the sender checks whether the resources are available before attempting to reserve it for use.

3.1.3.4 Delay in a Virtual Circuit Network

The total delay will be the sum of the setup delay, data transfer delay and teardown delay. The setup and teardown delay will be a one-time delay for a virtual circuit. The data transfer delay will vary depending on whether the resources are allocated on demand (waiting time at the switches) or during the set up phase (no delays at the switches).

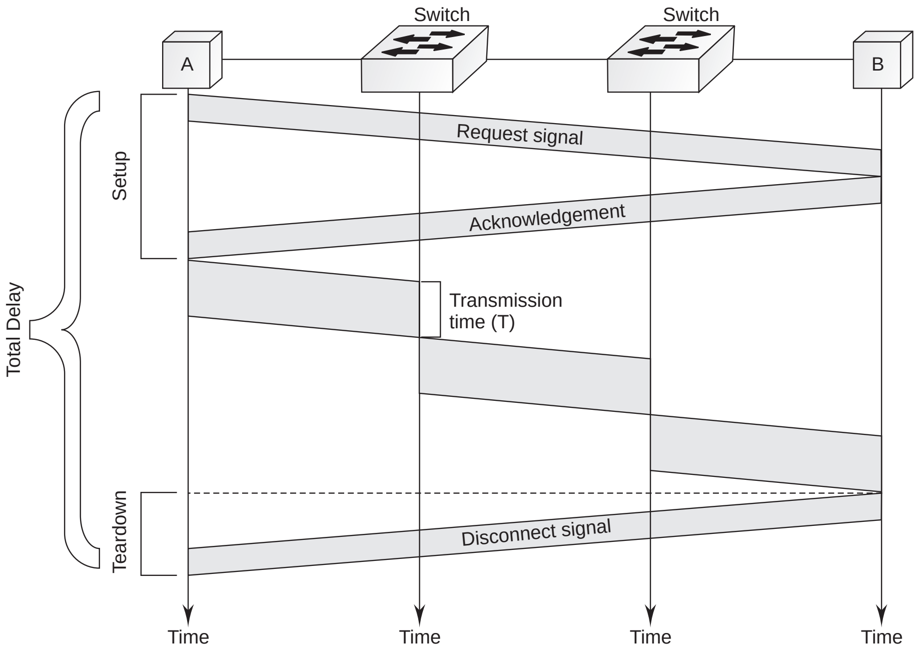
The following graph shows the delay for one packet of data to travel from sender A to receiver B via two switches. Let us take the example of resources being allocated at the setup phase thus there will be no waiting time at the switches.

Total delay would be equal to the sum of the setup delay, data transfer delay and finally the tear down delay.

The setup delay would include the setup request's propagation time $\mathbf{T_{rp}}$, the setup request signal's transfer time $\mathbf{T_{rt}}$, the acknowledgement signal's propagation time $\mathbf{T_{ap}}$, and **t**ransfer time of acknowledgement signal $\mathbf{T_{at.}}$

Similarly, the tear down delay would be the sum of the propagation $\mathbf{T_{tp}}$ and transfer delays $\mathbf{T_{tt}}$ of the teardown request in one direction.

The data transfer delay would be the sum of three transmission times **3T,** three propagation delays of **3τ.**



Total Delay = Setup delay + Data Transfer Delay + Teardown Delay

$$= T_{rp} + T_{rt} + T_{ap} + T_{at} + 3T + 3\,\tau\ + T_{tp} + T_{tt}$$

Figure 3.11: Calculation of total delay

3.1.4 Source Routing

A third approach to switching, that uses neither virtual circuits nor conventional datagrams, is known as source routing. The name derives from the fact that all the information about network topology that is required to switch a packet across the network is provided by the source host.

There are various ways to implement source routing. One would be to assign a number to each output of each switch and to place that number in the header of the packet. The switching function is then very simple: For each packet that arrives on an input, the switch would read the port n umber in the header and transmit the packet on that output.

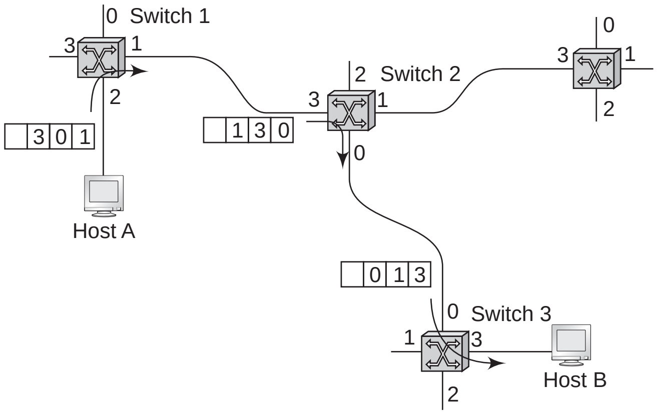


Figure 3.12: Source routing in a switched network
(where the switch reads the right most number)

However, since there will in general be more than one switch in the path between the sending and the receiving host, the header for the packet needs to contain enough information to allow every switch in the path to determine which output the packet needs to be placed on. One way to do this would be to put an ordered list of switch ports in the header and to rotate the list so that the next switch in the path is always at the front of the list. Figure 3.12 illustrates this idea.

In this example, the packet needs to traverse three switches to get from host A to host B. At switch 1, it needs to exit on port 1, at the next switch it needs to exit at port 0, and at the third switch it needs to exit at port 3.

Thus, the original header when the packet leaves host A contains the list of ports (3, 0, 1), where we assume that each switch reads the rightmost element of the list. To make sure that the next switch gets the appropriate information, each switch rotates the list after it has read its own entry. Thus, the packet header as it leaves switch 1 en route to switch 2 is now (1, 3, 0); switch 2 performs another rotation and sends out a packet with (0, 1, 3) in the header.

Although not shown, switch 3 performs yet another rotation, restoring the header to what it was when host A sent it.

There are several things to note about this approach. First, it assumes that host A knows enough about the topology of the network to form a header that has all the right directions in it for every switch in the path.

This is somewhat analogous to the problem of building the forwarding tables in a datagram network or figuring out where to send a setup packet in a virtual circuit network. Second, observe that we cannot predict how big the header needs to be, since it must be able to hold one word of information for every switch on the path.

This implies that headers are probably of variable length with no upper bound, unless we can predict with absolute certainty the maximum number of switches through which a packet will ever need to pass. Third, there are some variations on this approach.

For example, rather than rotate the header, each switch could just strip the first element as it uses it. Rotation has an advantage over stripping, however: Host B gets a copy of the complete header, which may help it figure out how to get back to host A.

Yet another alternative is to have the header carry a pointer to the current "next port" entry, so that each switch just updates the pointer rather than rotating the header; this may be more efficient to implement. We show these three approaches in Figure 3.13.

In each case, the entry that this switch needs to read is A, and the entry that the next switch needs to read is B.

Source routing can be used in both datagram networks and virtual circuit networks.

For example, the Internet Protocol, which is a datagram protocol, includes a source route option that allows selected packets to be source routed, while the majority is switched as conventional datagrams.

Source routing is also used in some virtual circuit networks as the means to get the initial setup request along the path from source to destination.

Finally, we note that source routing suffers from a scaling problem. In any reasonably large network, it is very hard for a host to get the complete path information it needs to construct correct headers.

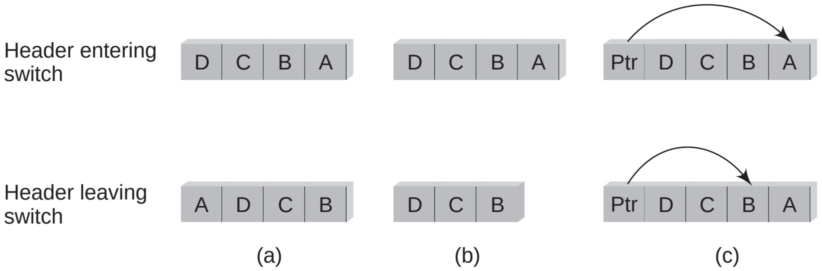


Figure 3.13: Three ways to handle headers for source routing: (a) rotation; (b) stripping: (c) pointer. The labels are read right to left

3.2 BRIDGES

A bridge operates in both physical and data link layer of OSI model. When a bridge acts as a physical layer device it regenerates the signal it receives. As a data link layer device, the bridge can check the physical (MAC) addresses, which are source and destination in the frame. Like repeaters bridges are used to connect to LAN segments. They retransmit packets from one segment on other segments. Bridges store and forward complete frames.

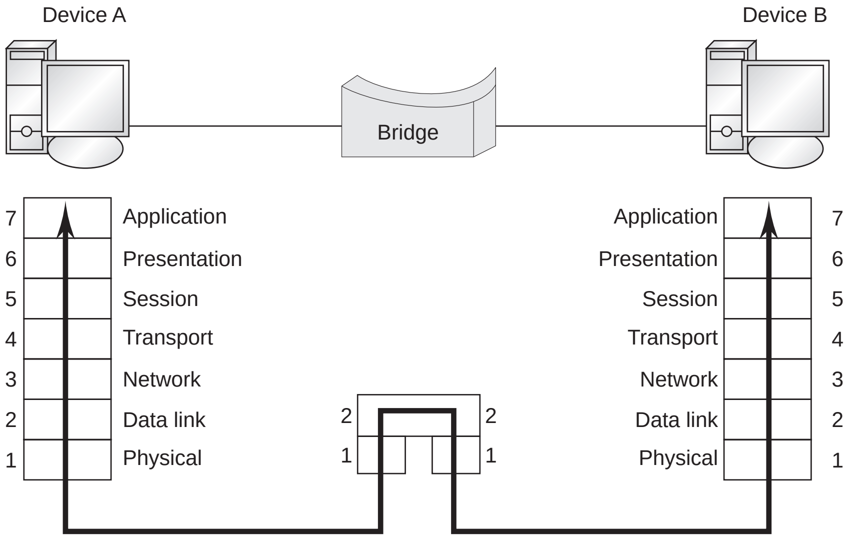


Figure 3.14: Position of bridges in OSI model

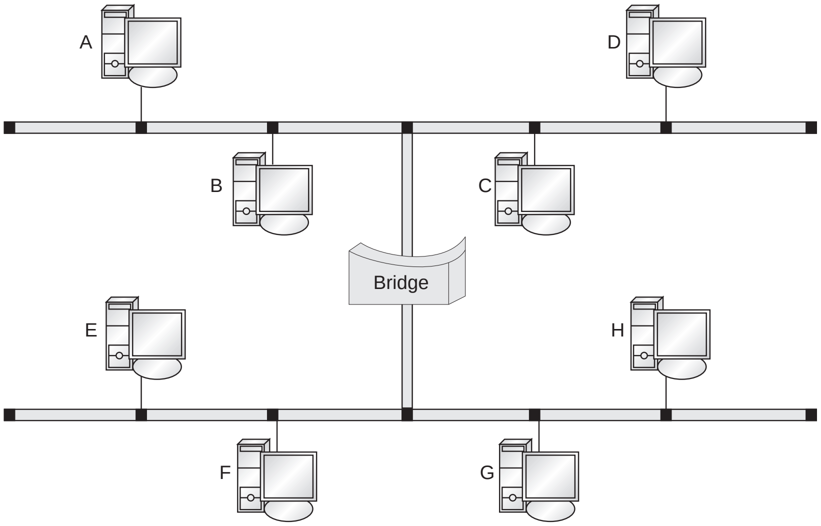
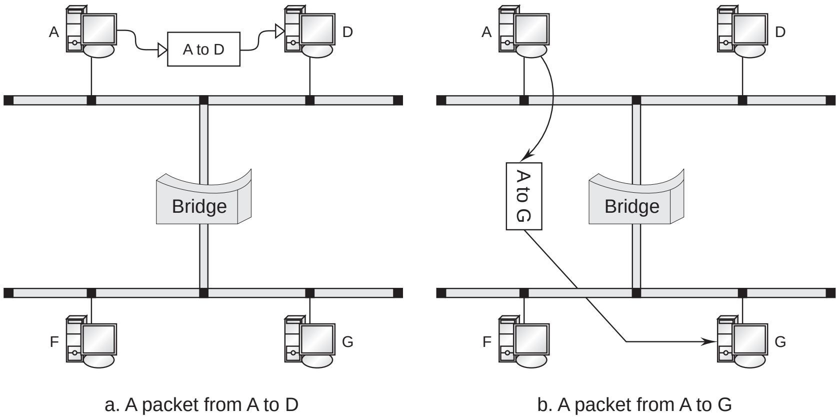


Figure 3.15: Bridge for a LAN

A bridge, by definition, should be able to connect LANs using different protocols at the data link layer, such as an Ethernet LAN to a wireless LAN. In bridges *store and forward* means, buffering all bits until the frame has been completely received. Bridges will check to make sure the frame is correct. It will transmit the frame on all interfaces, except the incoming interface. Bridges are invisible to all computers.



a. A packet from A to D

b. A packet from A to G

Figure 3.16: Data transfer across a bridge

The two main categories of bridges are local bridges or remote bridges. Local bridges provide a direct connection between multiple LAN segments in the same area. Remote bridges connect multiple LAN segments in different areas, usually over telecommunications lines.

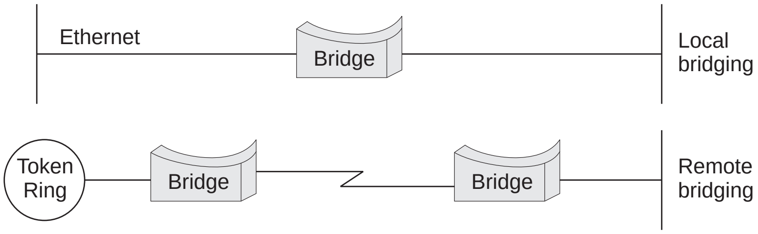


Figure 3.17: Category of bridges

3.2.1 Local Bridges

Local Bridges are used where the network is being locally segmented. The 2 segments are physically close together: same building, same floor, etc... Only 1 bridge is required. There are 3 primary bridging methodologies used by bridges for connecting local area networks:

- Transparent bridges
- Spanning Tree Protocol
- Source Routing

3.2.1.1 Transparent Bridges

Transparent Bridges examine the MAC address of the frames to determine whether the packet is on the local Segment or on the distant Segment. In transparent bridge the stations are completely unaware of the bridge's existence. If a transparent bridge is added or removed from the system, reconfiguration of the stations is not needed.

3.2.1.2 Spanning Tree Protocol

This is a link management protocol, which provides path redundancy and prevents undesirable loops in the network. There should be only one path exist between two stations for the smooth function of Ethernet network. Multiple active paths between stations cause loops in the network. When loops occur, some switches see stations appear on both sides of the switch. This condition confuses the forwarding algorithm and allows duplicate frames to be forwarded. To give path redundancy, Spanning-Tree Protocol defines a tree that spans all switches in an extended network. Spanning-Tree Protocol forces certain redundant data paths into a standby state.

3.2.1.3 Source Routing

In this technique bridge makes a routing decision based on the contents of the media access control (MAC) frame header of the frame. This header consists of a routing information field, which immediately follows the source address field in the frame. The routing information field describes the path across one or more bridges to the ring containing the destination station. With source routing bridges, the source and destination addresses of the frame on the media are the MAC addresses of the originating and target stations, respectively.

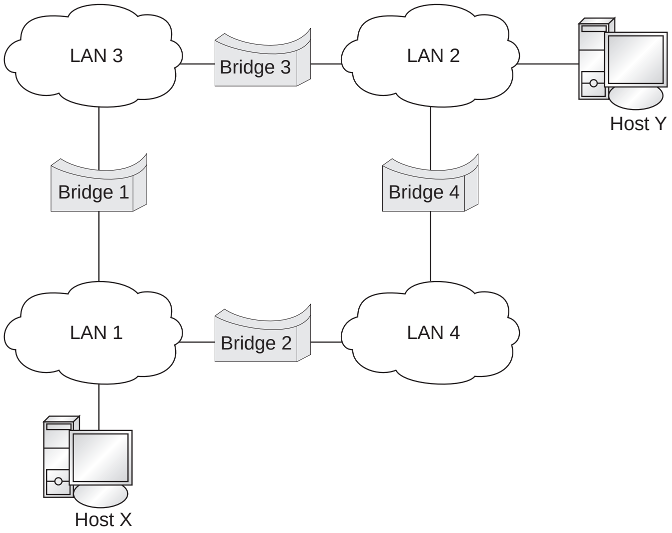


Figure 3.18: Source routing bridges

As given in the figure below, assume that Host X wants to send a frame to Host Y. At first, Host X does not know whether Host Y exists in on the same LAN or a different LAN. To determine this, Host X sends out a test frame. If that frame returns to Host X without a positive indication that Host Y has seen it, Host X assumes that Host Y is on a remote segment.

3.2.2 Remote Bridges

A basic use of bridges is to connect two or more remote LANs together. For example a corporation might have branches in many cities, each with its own LAN. All the LANs should be interconnected and the complete system acts like one large LAN. Since many enterprises are widely distributed, the LANs must often be connected with wide area communications links. These links connect LAN bridges as a point-to-point topology. This type of connection is called remote bridging.

Some call bridges as half bridges in the sense that two bridges and the link are considered to be a single bridge. Spanning tree operations can be applied to remote bridges. The point-to-point link is considered to be part of the spanning tree, and the bridges are forced to forward traffic on that link to the other bridge, this is the advantage. The disadvantage of this is that the IEEE in its initial discussions on spanning tree bridges, did not define fully remote bridge operations.

Another important fact is that if a LAN is connected through bridges into WAN topologies, these WANs will not provide the broadcasting capability. Therefore, it may be necessary for disbursed LANs to have their bridges fully engaged in order for the bridges to communicate with each other. This fully meshed network, while expensive, allows each designated LAN bridge to communicate with the other dedicated LAN bridges.

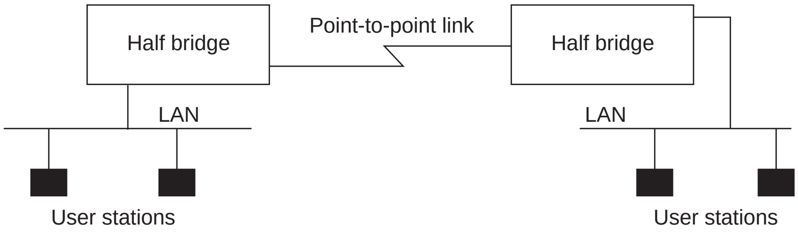


Figure 3.19: Remote bridges

3.2.2.1 Spanning Tree Algorithm

The Spanning Tree Protocol (STP) is a link layer network protocol that ensures a loop-free topology for any bridged LAN. Thus, the basic function of STP is to prevent bridge loops and ensuing broadcast radiation.

In the OSI model for computer networking, STP falls under the OSI layer 2. It is standardized as 802.1D. As the name suggests, it creates a spanning tree within a mesh network of connected layer-2 bridges (typically Ethernet switches), and disables those links that are not part of the spanning tree, leaving a single active path between any two network nodes.

Spanning tree allows a network design to include spare (redundant) links to provide automatic backup paths if an active link fails, without the danger of bridge loops, or the need for manual enabling/disabling of these backup links. Bridge loops must be avoided because they result in flooding the local network.

3.2.2.2 Protocol Operation

The collection of bridges in a LAN can be considered a graph whose nodes are the bridges and the LAN segments (or cables), and whose edges are the interfaces connecting the bridges to the segments.

To break loops in the LAN while maintaining access to all LAN segments, the bridges collectively compute a spanning tree.

The spanning tree is not necessarily a minimum cost spanning tree. A network administrator can reduce the cost of a spanning tree, if necessary, by altering some of the configuration parameters in such a way as to affect the choice of the root of the spanning tree.

Select a root bridge. The root bridge of the spanning tree is the bridge with the smallest (lowest) bridge ID. Each bridge has a unique identifier (ID) and a configurable priority number; the bridge ID contains both numbers. To compare two bridge IDs, the priority is compared first. If two bridges have equal priority, then the MAC addresses are compared. For example, if switches A (MAC=0200.0000.1111) and B (MAC=0200.0000.2222) both have a priority of 10, then switch A will be selected as the root bridge. If the network administrators would like switch B to become the root bridge, they must set its priority to be less than 10.

Determine the least cost paths to the root bridge. The computed spanning tree has the property that messages from any connected device to the root bridge traverse a least cost path, i.e., a path from the device to the root that has minimum cost among all paths from the device to the root. The cost of traversing a path is the sum of the costs of the segments on the path. Different technologies have different default costs for network segments. An administrator can configure the cost of traversing a particular network segment.

The property that messages always traverse least-cost paths to the root is guaranteed by the following two rules.

Least cost path from each bridge. After the root bridge has been chosen, each bridge determines the cost of each possible path from itself to the root. From these, it picks one with the smallest cost (a least-cost path). The port connecting to that path becomes the *root port* (RP) of the bridge.

Least cost path from each network segment. The bridges on a network segment collectively determine which bridge has the least-cost path from the network segment to the root. The port connecting this bridge to the network segment is then the *designated port* (DP) for the segment.

Disable all other root paths: Any active port that is not a root port or a designated port is a *blocked port* (BP).

Modifications in case of ties: The above rules over-simplify the situation slightly, because it is possible that there are ties, for example, two or more ports on a single bridge are attached to least-cost paths to the root or two or more bridges on the same network segment have equal least-cost paths to the root. To break such ties:

Breaking ties for root ports. When multiple paths from a bridge are least-cost paths, the chosen path uses the neighbor bridge with the lower bridge ID. The root port is thus the one connecting to the bridge with the lowest bridge ID. For example, in figure 3, if switch 4 were connected to network segment d, there would be two paths of length 2 to the root, one path going through bridge 24 and the other through bridge 92. Because there are two least cost paths, the lower bridge ID (24) would be used as the tie-breaker in choosing which path to use.

Breaking ties for designated ports. When more than one bridge on a segment leads to a least-cost path to the root, the bridge with the lower bridge ID is used to forward messages to the root. The port attaching that bridge to the network segment is the *designated port* for the segment.

In Figure 3.20, there are two least cost paths from network segment *d* to the root, one going through bridge 24 and the other through bridge 92. The lower bridge ID is 24, so the tie breaker dictates that the designated port is the port through which network segment d is connected to bridge 24. If bridge IDs were equal, then the bridge with the lowest MAC address would have the designated port. In either case, the loser sets the port as being blocked.

The final tie-breaker: In some cases, there may still be a tie, as when two bridges are connected by multiple cables. In this case, multiple ports on a single bridge are candidates for root port. In this case, the path which passes through the port on the neighbor bridge that has the lowest port priority is used.

The spanning tree that the bridges compute using the Spanning Tree Protocol can be determined using the following rules.

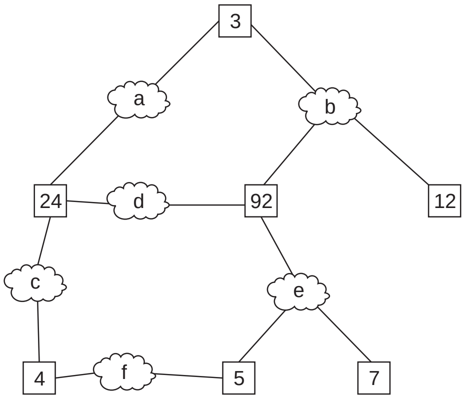
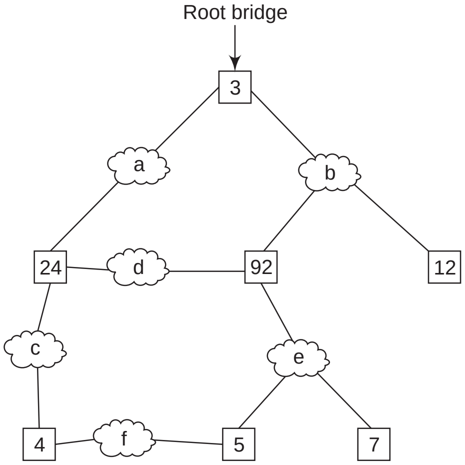
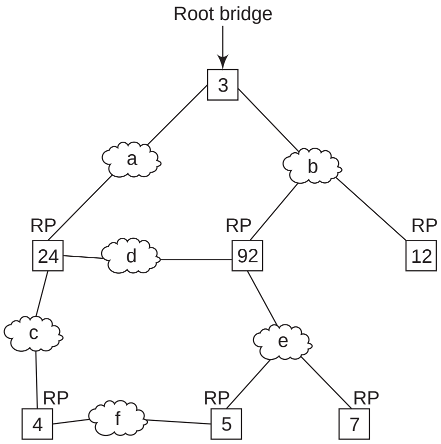


Figure 3.20: Spanning tree

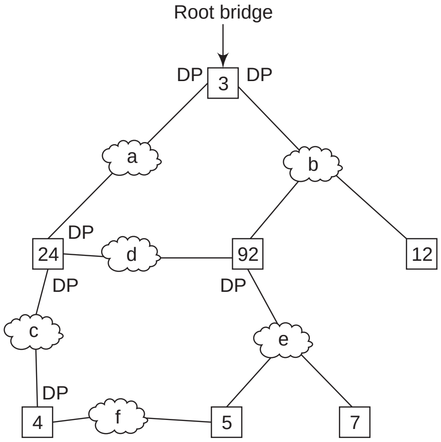
1. An example network. The numbered boxes represent bridges (the number represents the bridge ID). The lettered clouds represent network segments.



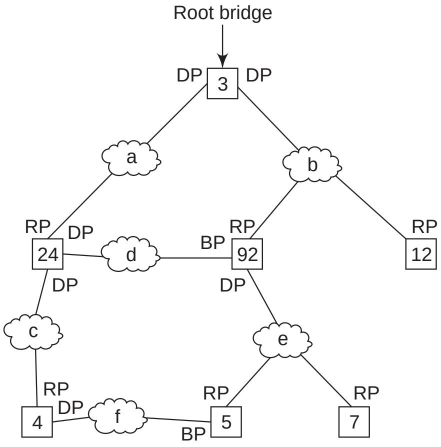
2. The smallest bridge ID is 3. Therefore, bridge 3 is the root bridge.



3. Assuming that the cost of traversing any network segment is 1, the least cost path from bridge 4 to the root bridge goes through network segment c. Therefore, the root port for bridge 4 is the one on network segment c.



4. The least cost path to the root from network segment *e* goes through bridge 92. Therefore the designated port for network segment e is the port that connects bridge 92 to network segment *e*.



5. This diagram illustrates all port states as computed by the spanning tree algorithm. Any active port that is not a root port or a designated port is a blocked port.

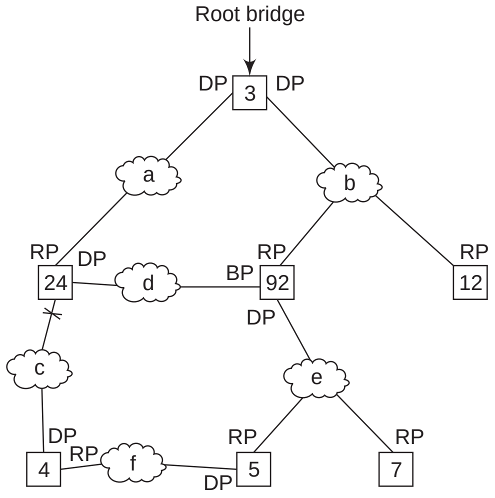


Figure 3.21: Port states computed by spanning tree algorithm

6. After link failure the spanning tree algorithm computes and spans new least-cost tree.

SUMMARY

- Switches basically change the direction of flow of data by making temporary connections between devices that have been connected to the switch. These interlinked nodes thus reduce the need to have as many links as required for a point-to-point connected network.
- Switching can be broadly classified into three categories:
  - o Circuit Switching
  - o Packet Switching
  - o Message Switching
- Packet switching can be further divided into:
  - o Datagram Networks
  - o Virtual Circuit networks
- Circuit-switched networks are those networks that establish a *dedicated* channel or circuit in advance between nodes and terminals.
- In circuit switching, when a station wants to set up a call or communicate with another station, the communication procedure can be divided into three phases:
  1. Establish circuit from end-to-end(“ dialing”) or Setup Phase
  2. Communicate or Data Transfer Phase
  3. Close circuit (“Teardown Phase”)
- Datagram switching, a sub-division of packet switching, is done at the *network layer.* Data is chopped up into packets before being sent over the packet-switched network. Each of these packets, even if it is just one piece of a huge multi-packet transmission, is considered as a separate entity in a datagram network. These packets are hereby referred to as ***datagrams.***

- Datagram networks are also known as ***connectionless networks*** as the switches do not retain information on the connection state.
- The job of the switch is to use the destination address as a key and perform a lookup on a data structure called a ***routing table***. This lookup returns an output port to forward the packet on towards the intended destination.
- Even though there are no setup or teardown phases in datagram switching, there is a considerable amount of delay which can be greater than the delay seen in a virtual-circuit network. The delay is due to the wait time that each packet will face at every switch it encounters in its journey towards the destination.
- *Virtual circuit packet switching* (VC-switching) is a packet switching technique which merges datagram packet switching and circuit switching to extract both of their advantages.
- The sending station tries to create a *virtual circuit* with the receiver. If this attempt is successful, then a different ***virtual circuit identifier (VCI)*** is allocated to each of the network devices taking part in the communication.
- Virtual Circuit networks require two types of addresses: **Global Address and Local address or Virtual Circuit Identifier (VCI)**
- The total delay in a virtual circuit network will be the sum of the setup delay, data transfer delay and teardown delay.
- Local Bridges are used where the network is being locally segmented.

4

INTERNET PROTOCOL

Introduction

Internetworking refers to the process of connecting individual computer networks or network segments together to form wide area networks (WANs) and connecting several WANs to create even larger WANs.

The science of internetworking can be quite complex as it is achieved using routers, bridges, and gateways and also interconnecting several networks that run on different protocols. The devices used to accomplish internetworking (switches, etc) are, as you might already know, layer 3 devices. While the physical and the data link layers of a network operate locally, it is the network layer (Layer 3) that helps guide the traffic along the *correct* path to its final destination. It is the network layer that is solely responsible for the delivery of packets from host-to-host and for routing the packets through the several routers and switches in the internetwork.

By just connecting networks together using bridges does not qualify it to be an internetwork, as it has no internetworking protocol like IP. Instead, it just makes up a sub-network. To convert a network into an internetwork, segregate the network into segments and then throw in routers or other layer 3 devices in between the segments. A perfect example of an internetwork is the Internet, where there are several networks with different protocols and there is the one unifying protocol (Internet Protocol IP).

4.1 INTERNET PROTOCOL (IP)

Each computer on the *Internet* has a unique numerical address, called an Internet Protocol (IP) address, used to route packets to it across the Internet. The Internet Protocol Suite consists of a set of communication protocols that the Internet and many other commercial networks run on. It is the most widely used open-system (or proprietary) protocol suite that can be used for communication across any set of Internetworks. This protocol suite is also referred to as the ***TCP/IP protocol suite*** after its two best-known communication protocols - Transmission Control Protocol (TCP) and Internet Protocol (IP).

The protocol suite consists of the lower layer protocols (TCP, IP) and even specifies common applications like e-mail, file transfer, and terminal emulation. This suite can be looked at as a set of layers, with each layer being involved in the transmission of data. It was developed in the 1970s when the US Department of Defense Advanced Research Projects Agency (DARPA) wanted heterogeneous connectivity (dissimilar computers and networks being able to communicate with each other).

The following diagram illustrates how the various protocols in the Internet protocol suite are mapped with the OSI layer model.

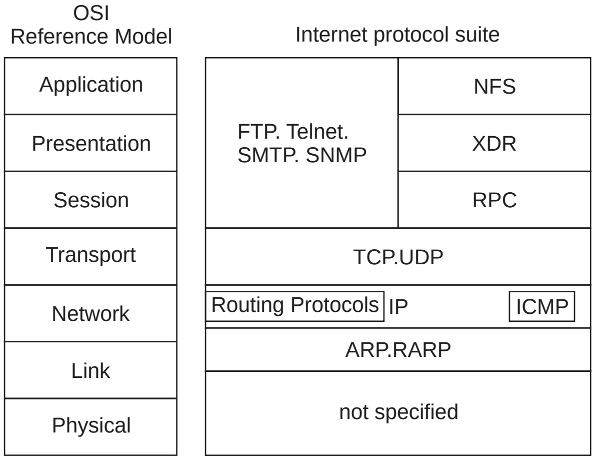


Figure 4.1: Mapping IP suite with OSI layer model

The ***Internet Protocol (IP)*** is the primary network layer protocol used for data communication across packet-switched internetworks. This data-oriented protocol gives unique global addressing information to computers and control information, all of which enable the routing of packets till its destination. Since the communicable addressing is done by IP, the data link layer protocol (like Ethernet) will only be concerned with the addressing till the next device. On the other hand, the IP addressing will ensure that the packet reaches its final destination. The Internet Protocol has basically two main roles:

1. Ensure connection-less transfer of data packets across the network
2. Fragment and reassemble the packets as per the data links requirement of maximum transmission unit (MTU) sizes.

4.1.1 IP Packet Format

An IP packet consists of an IP header followed by the data that can be of variable length. There are fourteen fields in an IP packet. A brief summary of each of these fields are given below:

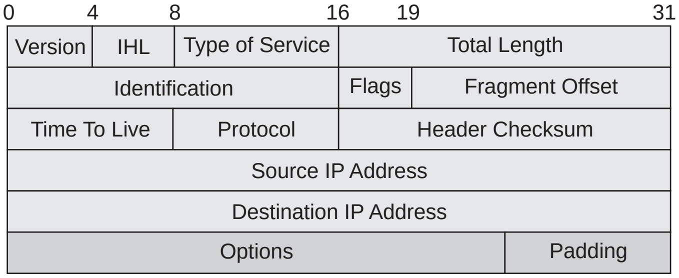


Figure 4.2: IP packet format

1. **Version (4 bits):** This field will specify the IP version.
2. **IHL (4 bits):** The Internet Header Length indicates the length of the header in 32-bit words. Minimum value is 5 for an accurate header. IHL would therefore point to the beginning of the data.
3. **Type of Service (8 bits):** The type of service field indicates the service parameters to be considered when transmitting the datagram across a specific network. It specifies how the datagram under consideration should be handled (normal/high reliability, normal/high delay etc) and also allocates priority levels to the datagrams. Certain networks provide service precedence, where datagrams of high precedence are given more significance when compared to other traffic. But, mostly, it is a compromise between low delay, high reliability, and high throughput. The following diagram shows the Type of Service field in a more detailed bit-by-bit cross-examination.

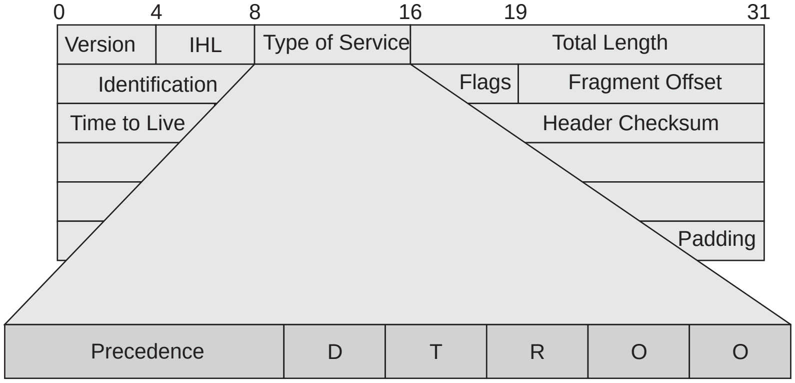


Figure 4.3: Description of type of service bits

Bits 0-2: Precedence.

Bit 3: 0 = Normal Delay, 1 = Low Delay.

Bit 4: 0 = Normal Throughput, 1 = High Throughput.

Bit 5: 0 = Normal Reliability, 1 = High Reliability.

Bit 6-7: Reserved for Future Use.

Precedence

111 - Network Control
110 - Internetwork Control
101 - CRITIC/ECP
100 - Flash Override

011 - Flash
010 - Immediate
001 - Priority
000 - Routine

Understandably, using the delay, throughput, and reliability indicators would hike the cost of the service. It is best that at the most two of the three indicators be set so that there's a balance between the three.

It is up to each network to use the network control precedence designation because it is to be used within a network only, specifically for the gateway control originators.

4. **Total Length (16 bits):** Indicates the length of the datagram (in octets), including the header and data. The maximum possible length of the IP packet can thus be 65,535 octets. This length is not practical for every host and network. Therefore, it was decided that all hosts should be able to receive a nominal sized datagram of 576 octets (data block – 512 octets, header-64 octets). Datagrams of size greater than 576 octets should be sent only if the sender is sure that the destination has the capacity to receive a large datagram.

5. **Identification (16 bits):** The integer in this field (assigned by the sending host) is an identifier for the datagram that will help in the final assembly of the datagram fragments.
6. **Flags (3 bits):** This field controls whether the packet can be fragmented and whether this datagram is the last in the series of fragmented packets or if there are more on their way. The high-order bit is not used and is zero.

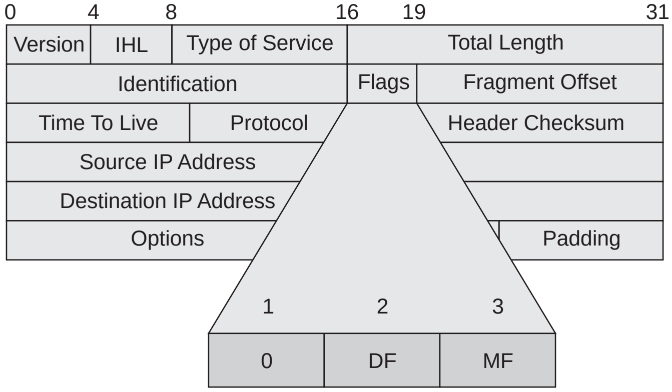


Figure 4.4: Valves for flag bits

Bit 0 : reserved, must be zero

Bit 1 : (DF) 0 = May Fragment, 1 = Don't Fragment.

Bit 2 : (MF) 0 = Last Fragment, 1 = More Fragments.

—X More

-X- don't fragment

X—Unused

7. **Fragment Offset (13 bits):** Indicates where the data in this fragment should be with reference to the beginning of the data in the original datagram. This helps in the reconstruction of the original datagram at the destination. It is measured in units of 8 octets (64 bits). This would mean that the first fragment's offset would be zero.
8. **Time to Live (8 bits):** Contains the maximum time in seconds that a datagram can remain in the internet system. This counter is decremented each time the datagram is processed by a module and when it reaches zero, it is discarded. TTL ensures that datagrams that cannot be delivered are discarded and do not loop endlessly in the network.
9. **Protocol (8 bits):** Specifies the upper-layer protocol that will receive the datagram after the packet has undergone IP processing.
10. **Header Checksum (16 bits):** This field recalculates and verifies some header fields (like TTL), which may change in the course of transmission, at every point that the IP header is processed.
11. **Source IP Address (32 bits):** Contains the 32-bit address of the sending node.
12. **Destination IP Address (32 bits):** Contains the 32-bit address of the destination node.
13. **Options (variable):** This field enables the support of several options, like security.
14. **Padding:** The internet header padding is used to ensure that the internet header ends on a 32 bit boundary. The padding is zero.

4.1.2 IP Addressing

The network layer communication (host-to-host) requires a global addressing scheme that is necessary to route IP datagrams across a network. An ***IP address*** is the term given for a logical address in the network layer of the TCP/IP protocol suite. It has a basic format made up of components divided by decimal dots. It is possible to subdivide the IP address further and create addresses for sub-networks.

Each host on a TCP/IP network is given a unique IP address that identifies only one connection to the Internet. The Internet addresses are 32 bits in length (IPv4 addresses), thus a maximum of 2^{32} addresses (or 4,294,967,296- more than 4 billion addresses) can be assigned. No two devices can have the same address at the same time but it is possible that an address can be given to one host for a given period of time and then given to some other host.

Every IP address consists of two parts, one identifying the network that the computer belongs to known as the Network ID or Network Address (The number assigned to a single network in an internetwork. Network addresses are used by hosts and routers when routing a packet from a source to a destination in an internetwork) and one identifying the host on the network known as the Host ID or Host Address (can be either be the host's physical address (the address of the network interface card) or an administratively assigned address that uniquely identifies the host on its network).

The network ID has to be assigned by the Internet Network Information Center (InterNIC) if the network is to be part of the Internet. Mostly, Internet Service Providers (ISP) takes blocks of addresses from the InterNIC and then assigns the addresses to its customers as per requirement. The host number, on the other hand, is assigned by the local network administrator. The combination of the network address and the host address; it uniquely identifies a host on an internetwork is known as Internetwork address.

4.1.3 IP Address Format

An IP address is made up of four bytes of information (totaling 32 bits) expressed as four numbers between 0 and 255 shown separated by periods. The IP address, a 32 bit binary number, is usually represented as 4 decimal values, with 8 bits being grouped at a time. The four octets are separated by decimal dots and this format of the IP address is known as the ***dotted decimal notation.*** Since each of the 8 positions can have two different states (0 or 1), the total number of combinations per octet is 2^8 or 256. This means that the minimum value for an octet is 0 while the maximum value is 255.

The basic IP address format is shown in the following diagram:

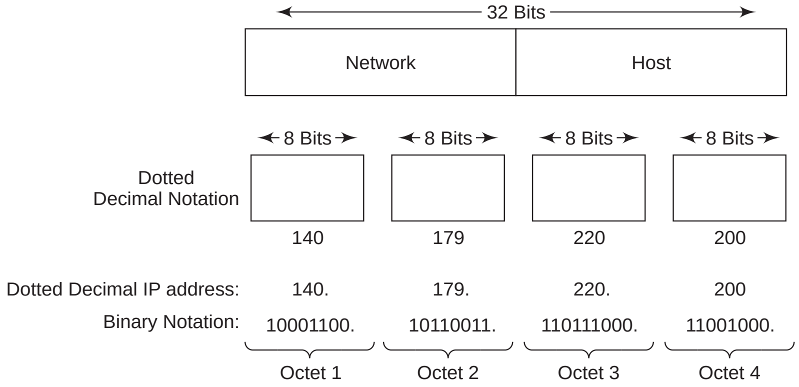


Figure 4.5: IP address format

For example, your computer's IP address might be 238.17.159.4, which is shown below in human-readable decimal form and in the binary form used on the Internet.

Example IP Address	
Decimal:	238 .17 .159. 4
Binary:	11101110 00010001 10011111 00000100

Each of the four numbers uses eight bits of storage, and so can represent any of the 256 numbers in the range between zero (binary 00000000) and 255 (binary 11111111). Therefore, there are more than 4 billion possible different IP addresses in all:

4,294,967,296 = 256 * 256 * 256 * 256

4.1.4 IP Address Classes

In the original IP addressing, the concept of classes was used. This concept is fast becoming obsolete but we will anyway briefly discuss the meaning of ***classful addressing.*** The address space was divided into five classes: A, B, C, D, and E. Out of these, only classes A, B, and C are used for commercial purposes. These address classes differ in size and number. If an address is given in its binary or dotted-decimal form, you can determine the class it belongs to by just observing the first few bits (for binary notation) or the first byte (for decimal notation) in the address.

To understand which class an address would belong to, note the left-most or higher bytes of an address as shown in the following figure:

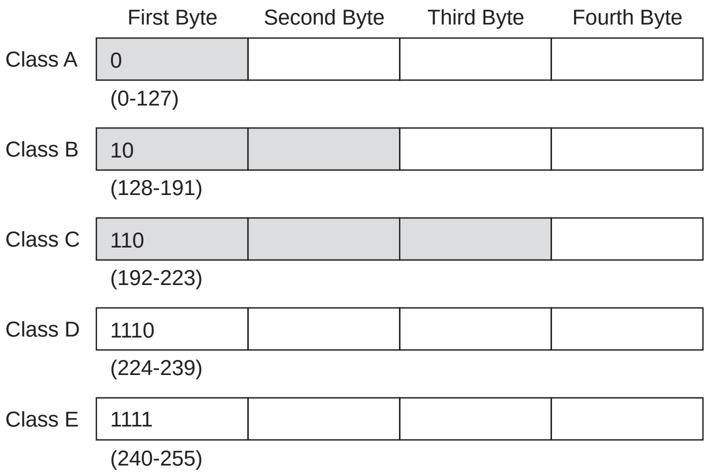


Figure 4.6: IP address classes

The binary notation is written within the boxes while the decimal equivalent is written below the first byte box. Class D addresses are kept aside for multicast purposes while Class E addresses are reserved for future use.

Let us take up some examples and find out which address class it belongs to:

- **16**.0.0.5: The first byte 16 (between 0 and 126, hence it belongs to Class A)
- **110**01111 00000011 11000000 11111111: The first two bits are 11 and the third one is 0, which means it is a Class C address.

Netid and Hostid

Depending on the class of address, an IP address is divided into the ***netid*** and ***hostid***. Its length varies from class to class. The *netid* would represent the network the host belongs to, while the *hostid* would be its unique id within the network.

The following figure shows how many bits are kept aside for the network part and host part for classes A, B, and C.

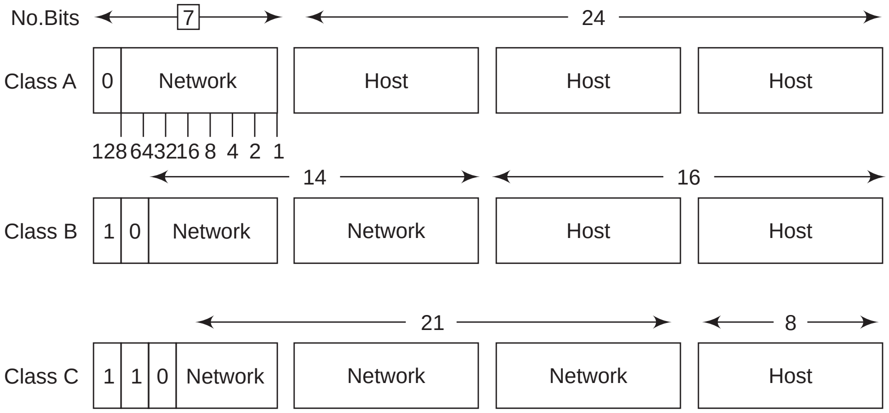


Figure 4.7: Division of IP address into network and host part

Just one byte defines the *netid* for Class A, while in Class B, two bytes is kept aside for the network id. Class C, on the other hand, has just one byte kept aside for its *hostid*. So, when an IP packet is being routed using internet routing, and it reaches a router, it first extracts the destination address from the packet. This address is examined and its class is determined.

Once this is done, it is dissected into the network and host ids. The router only looks at the network id so that it routes the packet to the right network. Once the packet reaches its destination network, the *hostid* is taken into consideration so that it is finally sent to its destination computer.

Keeping all these facts in mind, the following table shows a summary of the address classes with their address ranges:

Table 4.1: Address classes

IP Addresses	Format	Purpose	High-order bits	Address Range	Number of bits (Network/Host)	Max. Hosts
A	N.H.H.H	Few large organizations	0	1.0.0.0 to 126.0.0.0	7/24	16777214 (2^{24}-2)
B	N.N.H.H	Medium sized organizations	1,0	128.1.0.0 to 191.254.0.0	14/16	65534 (2^{16}-2)
C	N.N.N.H	Relatively small organizations	1,1,0	192.0.1.0 to 223.255.254.0	21/8	254 (2^{8}-2)
D	N/A	Multicast groups	1,1,1,0	224.0.0.0 to 239.255.255.255	Not for commercial use	N/A
E	N/A	Experimental	1,1,1,1	240.0.0.0 to 254.255.255.255	N/A	N/A

4.1.4.1 Subnetting and Subnet Mask

IP Subnet Addressing

Subnetting is the process by which the IP networks were further divided into smaller networks or subnets. This gave network administrators a lot of flexibility when assigning addresses to the computers. It also made a more efficient use of network addresses (which were otherwise being wasted), and the ability to broadcast traffic.

The presence of subnets would be hidden from the outside world as it sees it as just a single network. How the network is internally structured is known only to the computers within the network. A single network address can be divided into several subnetworks like 192.168.1.0 being divided into 192.168.1.5, 192.168.1.6, etc.

IP Subnet Mask

Deciding the subnet address is done by taking some bits from the already allocated hostid and then assigning them to the subnet field. Now, how many of these bits are taken will depend on the *subnet mask.* The following figure shows how some bits from the hostid in a Class B address is kept aside as the subnet field.

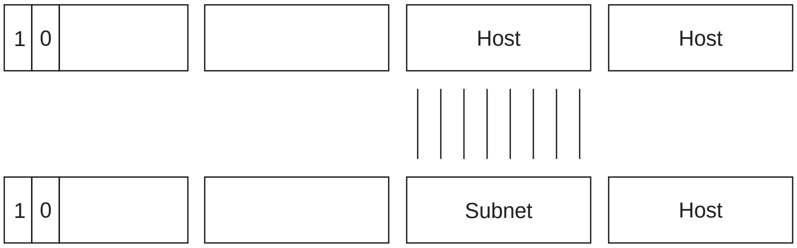


Figure 4.8(a): Class B address: After subnetting

To write the subnet mask, enter 1s in the place of the network and subnet fields and 0s in the host field. The following figure shows a sample subnet mask.

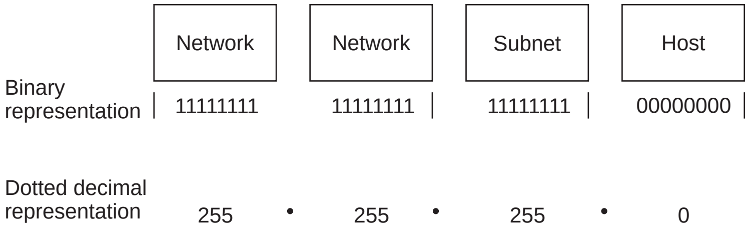


Figure 4.8(b): Sample subnet mask

When writing the subnet mask, choose the bits from the left-most side of the host field. The next table shows subnet mask values when one does not choose the entire byte to be 1s.

If there is no subnetting for a Class B address, the subnet mask would be 255.255.255.0. Take a class B address like 190.12.0.0. If eight bits are to be taken for subnetting, then the subnet mask would be 255.255.255.0.

128 ↓	64 ↓	32 ↓	16 ↓	8 ↓	4 ↓	2 ↓	1 ↓		
1	0	0	0	0	0	0	0	=	128
1	1	0	0	0	0	0	0	=	192
1	1	1	0	0	0	0	0	=	224
1	1	1	1	0	0	0	0	=	240
1	1	1	1	1	0	0	0	=	248
1	1	1	1	1	1	0	0	=	252
1	1	1	1	1	1	1	0	=	254
1	1	1	1	1	1	1	1	=	255

Then, number of possible subnets = $2^{\text{no. of bits for subnetting}} - 2$
$= 2^8 - 2$
= 254 subnets.

The number 2 is reduced because one address is to be kept for a network address while one if for the broadcast address.

Number of hosts per subnet = $2^{\text{no. of bits for subnetting}} - 2$
$= 2^8 - 2$
= 254 hosts per subnet

Now, take a Class C address 192.168.1.0 with three bits of subnetting. Then, the subnet mask would be 255.255.255.224.

Number of possible subnets = $2^3 - 2$
= 6 subnets.

Number of hosts per subnet = $2^5 - 2$
= 30 hosts per subnet.

4.1.4.2 Determining the Network Number using the Subnet Masks

When an IP packet reaches the router, it first obtains the destination address and retrieves the internal subnet mask. It then logically ANDs them together and finally gets the network number. By doing this, the host field is removed and only the network field remains. Thus, it can now route the packet using the destination network number on to its outgoing interface. The packet will be now on its way to its destination.

The subnet mask is used to obtain the subnetwork number in the following example. Since the IP address (200.23.34.77) belongs to Class C, its subnet mask would be 255.255.255.0.

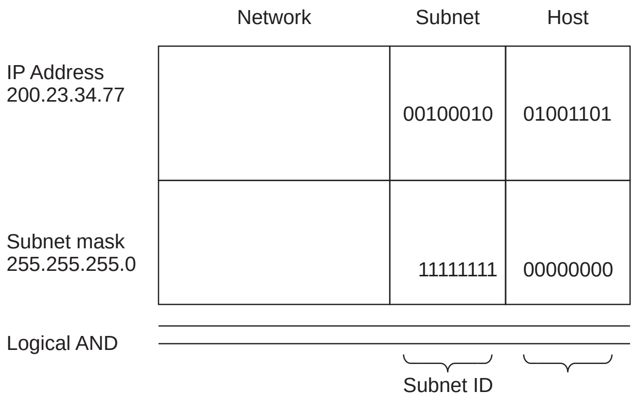


Figure 4.9: Calculating network number using subnet mask

4.2 ROUTING

The process of finding a path from source to destination is called routing. Routing can be characterized based on when the route is actually established between any two nodes on the network. Routing can thus be of two types:

Static: The path between two nodes is decided during the design of the network itself. These paths are stored in a *routing table*. When the packet has to be transferred, the routing table gives the route to be taken by the packet. If the network is fixed then static routing algorithm will work fine but it fails to identify any alternate routes if any node has been removed from the network.

Dynamic: It decides the path followed by the packet at run time i.e. when the packet is being sent. It continually tracks for changes in the network configuration, so that the route can be modified in case any node has been removed or a new node has been added.

4.2.1 Need for Routing

A network is an interconnected set of nodes (with a network wide system of unique addresses). In general there may be several different routes from one node to another. There is no requirement for a direct connection between any two nodes. Any given node will have a direct connection to one or more other nodes.

If a node receives a packet addressed to a directly connected node, it will simply pass it to the appropriate link driver software. If a node receives a packet addressed to a node to which it has no direct connection, it must solve the “routing problem” to determine which node to send it to.

A typical network is illustrated below.

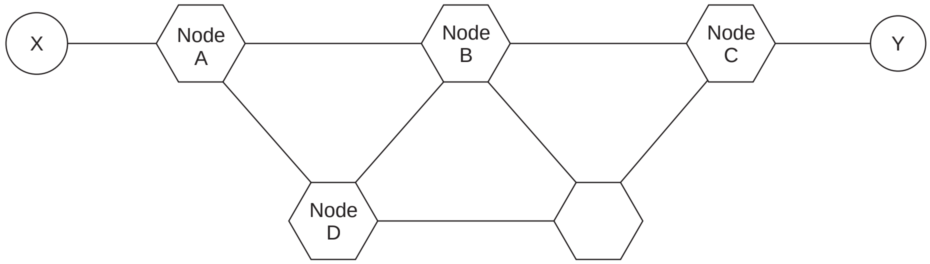


Figure 4.10: Typical routed network

Node A may route a packet addressed to Y to either node B or to node D. The selection of which node to route the packet to, is known as the *routing decision*. The routing decision is made by an algorithm incorporated in the packet switching exchange (PSE) software of node A.

The performance of routing algorithms may be expressed in terms of various criteria.

- o Performance Criteria
  - Minimum Number of Hops
  - Minimum "cost"
  - Minimum delay
  - Maximum throughput
  - Minimum CPU processing at nodes
  - Minimum CPU memory requirement at nodes
- o Decision Time
  - Each Packet (Datagram)
  - Session (Virtual Call)
  - Configuration (Permanent Virtual Circuit)
- o Decision Place
  - Each Node (Distributed)
  - Central
  - Originating Node
- o Information Source
  - None
  - Local
  - Adjacent Nodes
  - Nodes Along Route
  - All Nodes
- o Information Update
  - Continuous
  - Periodic
  - Major Load Change
  - Topology Change

A robust routing mechanism is one that copes with link failures and congestion. This can be achieved by an *adaptive strategy*. Such a strategy should avoid instability; a tendency to respond too rapidly or extremely to perceived changes in network performance.

Routing can be deterministic (which means routing tables) or non-deterministic (techniques that do not depend on network topology).

Routing Table- It contains the information required for routing. It contains the address of the node across the path and the cost between each pair of nodes.

Non-deterministic Routing

This is rarely used in practice but the following techniques have been proposed and discussed.

1. **Flooding:** A very simple technique, where copies of incoming packets are sent by **EVERY** link *except* the one from which the packets arrived. This generates an enormous amount of superfluous traffic.

   To prevent endless copies of packets circulating indefinitely through the network, a ***hop count*** may be used to suppress onwards transmission of packets after a number of hops exceeding the network "diameter". The destination must be prepared to receive multiple copies of an incoming packet.

   Flooding has two interesting characteristics that arise from the fact that all possible routes are tried :-

   (*a*) As long as there is a route from source to destination, the delivery of the packet is guaranteed

   (*b*) One copy of the packet will arrive by the quickest possible route Flooding is an extremely robust technique and would be particularly suitable for a "battlefield" situation. The second property might be useful for "route learning".

4.2.2 Distance Vector Routing

Here, each node constructs a one-dimensional array (a vector) containing the "distances" (costs) to all other nodes and distributes that vector to its immediate neighbors. The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors. A link that is down is assigned an infinite cost.

To see how a distance-vector routing algorithm works, it is easiest to consider an example like the one depicted in the following figure. In this example, the cost of each link is set to 1, so that a least-cost path is simply the one with the fewest hops. (Since all edges have the same cost, we do not show the costs in the graph).

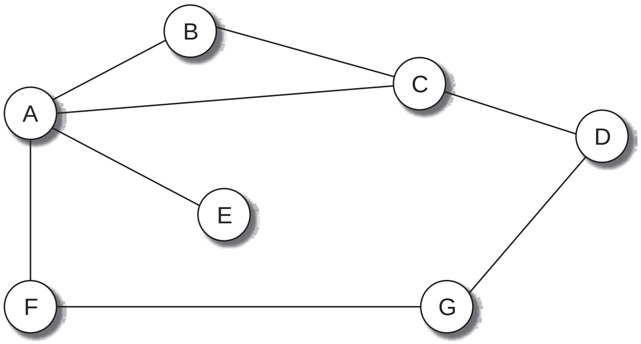


Figure 4.11: Distance-vector routing: an example network

We can represent each node's knowledge about the distances to all other nodes as a table like the one given in Table 4.5. Note that each node only knows the information in one row of the table (the one that bears its name in the left column). The global view that is presented here is not available at any single point in the network.

We may consider each row in the above table as a list of distances from one node to all other nodes, representing the current beliefs of that node. Initially, each node sets a cost of 1 to its directly connected neighbors and ∞ to all other nodes.

Table 4.2: Initial distances stored at each node (global view)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Thus, A initially believes that it can reach B in one hop and that D is unreachable. The routing table stored at A reflects this set of beliefs and includes the name of the next hop that A would use to reach any reachable node. Initially, then, A's routing table would look like the table below.

Table 4.3: Initial routing table at node A

Destination	Cost	Next Hop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

The next step in distance-vector routing is that every node sends a message to its directly connected neighbors containing its personal list of distances. For example, node F tells node A that it can reach node G at a cost of 1; A also knows it can reach F at a cost of 1, so it adds these costs to get the cost of reaching G by means of F.

This total cost of 2 is less than the current cost of infinity, so A records that it can reach G at a cost of 2 by going through F. Similarly, A learns from C that D can be reached from C at a cost of 1; it adds this to the cost of reaching C (1) and decides that D can be reached via C at a cost of 2, which is better than the old cost of infinity.

At the same time, A learns from C that B can be reached from C at a cost of 1, so it concludes that the cost of reaching B via C is 2. Since this is worse than the current cost of reaching B (1), this new information is ignored.

At this point, A can update its routing table with costs and next hops for all nodes in the network. The result is shown in the table below.

In the absence of any topology changes, it only takes a few exchanges of information between neighbors before each node has a complete routing table. The process of getting consistent routing information to all the nodes is called *convergence*.

Table 4.4: Final routing table at node A

Destination	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

The table below shows the final set of costs from each node to all other nodes when routing has converged. We must stress that there is no one node in the network that has all the information in this table—each node only knows about the contents of its own routing table. The beauty of a distributed algorithm like this is that it enables all nodes to achieve a consistent view of the network in the absence of any centralized authority.

Table 4.5: Final distances stored at each node (global view)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

There are a few details to fill in before our discussion of distance-vector routing is complete. First we note that there are two different circumstances under which a given node decides to send a routing update to its neighbors.

One of these circumstances is the *periodic* update. In this case, each node automatically sends an update message every so often, even if nothing has changed. This serves to let the other nodes know that this node is still running. It also makes sure that they keep getting information that they may need if their current routes become unviable.

The frequency of these periodic updates varies from protocol to protocol, but it is typically on the order of several seconds to several minutes. The second mechanism, sometimes called a *triggered* update, happens whenever a node receives an update from one of its neighbors that causes it to change one of the routes in its routing table.

That is, whenever a node's routing table changes, it sends an update to its neighbors that may lead to a change in their tables, causing them to send an update to their neighbors.

Now consider what happens when a link or node fails. The nodes that notice first send new lists of distances to their neighbors, and normally the system settles down fairly quickly to a new state. As to the question of how a node detects a failure, there are a couple of different answers.

In one approach, a node continually tests the link to another node by sending a control packet and seeing if it receives an acknowledgment. In another approach, a node determines that the link (or the node at the other end of the link) is down if it does not receive the expected periodic routing update for the last few update cycles.

To understand what happens when a node detects a link failure, consider what happens when F detects that its link to G has failed. First, F sets its new distance to G to infinity and passes that information along to A.

Since A knows that its 2-hop path to G is through F, A would also set its distance to G to infinity. However, with the next update from C, A would learn that C has a 2-hop path to G. Thus A would know that it could reach G in 3 hops through C, which is less than infinity, and so A would update its table accordingly. When it advertises this to F, node F would learn that it can reach G at a cost of 4 through A, which is less than infinity, and the system would again become stable.

Unfortunately, slightly different circumstances can prevent the network from stabilizing.

Suppose, for example, that the link from A to E goes down. In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E. Depending on the exact timing of events, the following might happen: Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A; node A concludes that it can reach E in 4 hops and advertises this to C; node C concludes that it can reach E in 5 hops; and so on.

This cycle stops only when the distances reach some number that is large enough to be considered infinite. In the meantime, none of the nodes actually knows that E is unreachable, and the routing tables for the network do not stabilize. This situation is known as the *count-to-infinity* problem.

There are several partial solutions to this problem. The first one is to use some relatively small number as an approximation of infinity. For example, we might decide that the maximum number of hops to get across a certain network is never going to be more than 16, and so we could pick 16 as the value that represents infinity. This at least bounds the amount of time that it takes to count to infinity. Of course, it could also present a problem if our network grew to a point where some nodes were separated by more than 16 hops.

One technique to improve the time to stabilize routing is called *split horizon*. The idea is that when a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor. For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update. In a stronger variation of split horizon, called *split horizon with poison reverse*, B actually sends that route back to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E. For example, B sends the route (E, ∞) to A. The problem with both of these techniques is that they only work for routing loops that involve two nodes. For larger routing loops, more drastic measures are called for.

Continuing the above example, if B and C had waited for a while after hearing of the link failure from A before advertising routes to E, they would have found that neither of them really had a route to E. Unfortunately, this approach delays the convergence of the protocol; speed of convergence is one of the key advantages of its competitor, link-state routing, discussed next.

4.2.3 Link State Routing

Link-state routing is the second major class of intradomain routing protocol.

The starting assumptions for link-state routing are rather similar to those for distance vector routing.

Each node is assumed to be capable of finding out the state of the link to its neighbors (up or down) and the cost of each link. The basic idea behind link-state protocols is very simple:

Every node knows how to reach its directly connected neighbors, and if we make sure that the totality of this knowledge is disseminated to every node, then every node will have enough knowledge of the network to build a complete map of the network.

Thus, link-state routing protocols rely on two mechanisms:

- Reliable distribution of link-state information.
- Calculation of routes from the sum of all the accumulated link-state knowledge.

4.2.3.1 Reliable Flooding

Reliable flooding is the process of making sure that all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes.

As the term "flooding" suggests, the basic idea is for a node to send its link-state information out on all of its directly connected links, with each node that receives this information forwarding it out on all of its links. This process continues until the information has reached all the nodes in the network.

Each node creates an update packet, also called a link-state packet (LSP), that contains the following information:

- the ID of the node that created the LSP
- a list of directly connected neighbors of that node, with the cost of the link to each one
- a sequence number
- a time to live for this packet

The first two items are needed to enable route calculation; the last two are used to make the process of flooding the packet to all nodes reliable.

Flooding works in the following way. First, the transmission of LSPs between adjacent routers is made reliable using acknowledgments and retransmissions. However, there are several more steps needed to reliably flood an LSP to all nodes in a network.

Consider a node X that receives a copy of an LSP that originated at some other node Y. Note that Y may be any other router in the same routing domain as X. X checks to see if it has already stored a copy of an LSP from Y. If not, it stores the LSP. If it already has a copy, it compares the sequence numbers; if the new LSP has a larger sequence number, it is assumed to be the more recent, and that LSP is stored, replacing the old one. A smaller (or equal) sequence number would imply an LSP older (or not newer) than the one stored, so it would be discarded and no further action would be needed. If the received LSP was the newer one, X then sends a copy of that LSP to all of its neighbors except the neighbor from which the LSP was just received. The fact that the LSP is not sent back to the node from which it was received helps to bring an end to the flooding of an LSP. Since X passes the LSP on to all its neighbors, who then turn around and do the same thing, the most recent copy of the LSP eventually reaches all nodes.

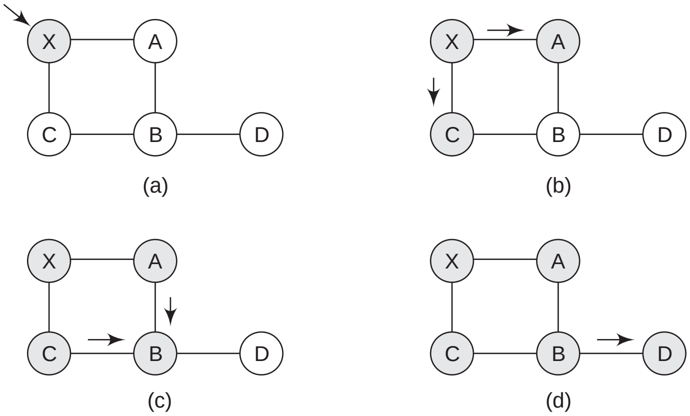


Fig 4.12 Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) and flood LSP to B (but not X); (d) flooding is complete.

Each node becomes shaded as it stores the new LSP. In the above figure the LSP arrives at node X, which sends it to neighbors A and C in Figure b. A and C do not send it back to X, but send it on to B. Since B receives two identical copies of the LSP, it will accept whichever arrived first and ignore the second as a duplicate. It then passes the LSP on to D, who has no neighbors to flood it to, and the process is complete.

Each node generates LSPs under two circumstances. Either the expiry of a periodic timer or a change in topology can cause a node to generate a new LSP. However, the only topology-based reason for a node to generate an LSP is if one of its directly connected links or immediate neighbors has gone down.

The failure of a link can be detected in some cases by the link-layer protocol. The demise of a neighbor or loss of connectivity to that neighbor can be detected using periodic "hello" packets.

Each node sends these to its immediate neighbors at defined intervals. If a sufficiently long time passes without receipt of a "hello" from a neighbor, the link to that neighbor will be declared down, and a new LSP will be generated to reflect this fact.

One of the important design goals of a link-state protocol's flooding mechanism is that the newest information must be flooded to all nodes as quickly as possible, while old information must be removed from the network and not allowed to circulate.

To make sure that old information is replaced by newer information, LSPs carry sequence numbers. Each time a node generates a new LSP, it increments the sequence number by 1. Unlike most sequence numbers used in protocols, these sequence numbers are not expected to wrap, so the field needs to be quite large (say, 64 bits).

If a node goes down and then comes back up, it starts with a sequence number of 0. If the node was down for a long time, all the old LSPs for that node will have timed out otherwise, this node will eventually receive a copy of its own LSP with a higher sequence number, which it can then increment and use as its own sequence number. This will ensure that its new LSP replaces any of its old LSPs left over from before the node went down.

LSPs also carry a time to live. This is used to ensure that old link-state information is eventually removed from the network. A node always decrements the TTL of a newly received LSP before flooding it to its neighbors. It also "ages" the LSP while it is stored in the node. When the TTL reaches 0, the node refloods the LSP with a TTL of 0, which is interpreted by all the nodes in the network as a signal to delete that LSP.

4.2.3.2 Route Calculation

Once a given node has a copy of the LSP from every other node, it is able to compute a complete map for the topology of the network, and from this map it is able to decide the best route to each destination.

The solution is based on a well-known algorithm from graph theory—Dijkstra's shortest-path algorithm

Dijkstra's Algorithm (Shortest Path)

The forward search algorithm can be described informally thus:

C(i,n) is the cost of the least cost path from i to n

l(i,n) is the link cost from i to n

```
for each node i
  for all other nodes set $C(i,n)$ to $L(i,n)$
  repeat
    find a node w (not yet considered by the algorithm) such
    that $C(i,w)$ is a minimum for all unconsidered nodes
    for each node n other than i and w
    do
      if $C(i,w)+L(w,n) < C(i,n)$
      then
        $C(i,n) = C(i,w)+L(w,n)$
         $path(i,n) = path(i,w)+path(w,n)$
      endif
    end do
    add the node w to the set of nodes considered so far
  until all nodes considered
```

Here's an example of a simple network showing how the *forward search* algorithm may be used.

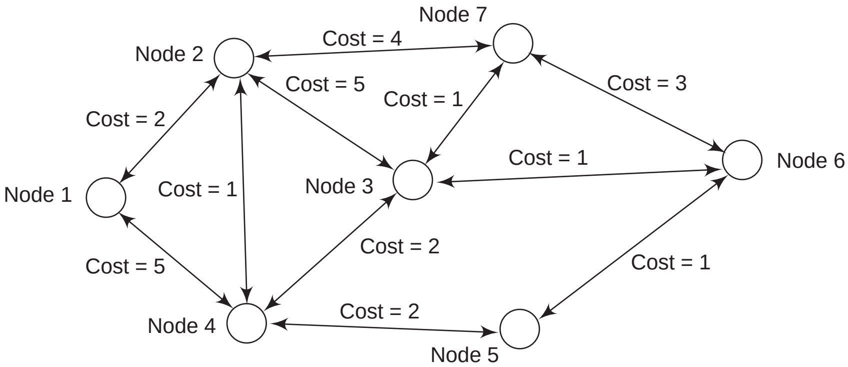


Figure 4.13: Network showing forward search algorithm

Here is the initial link cost table. Note that links costs are the same in each direction.

	1	2	3	4	5	6	7
1	0	2	Infinite	5	Infinite	Infinite	Infinite
2	2	0	5	1	Infinite	Infinite	4
3	Infinite	5	0	3	Infinite	1	Infinite
4	5	1	3	0	2	Infinite	Infinite
5	Infinite	Infinite	Infinite	2	0	1	Infinite
6	Infinite	Infinite	1	Infinite	1	0	3
7	Infinite	4	Infinite	Infinite	Infinite	3	0

The following notation is used

- C(i,n) - Cost of least cost path from node *i* to node *n*
- L(i,n) - Link cost from node *i* to node *n*. This has the value *Infinite* if there is no direct connection

Here is how the forward search algorithm proceeds for node 1.

1. **Initial State**

Node	2	3	4	5	6	7
Least cost path	2	Infinite	5	Infinite	Infinite	Infinite
Route	1-2		1-4			

2. Identification of least cost "unconsidered" node, identifies node 2. Examination of possible routes to other nodes proceeds thus.

Node	Formula	Value	Comparison	Result
3	C(1,2)+L(2,3)	2+5=7	7 < Infinite	New route
4	C(1,2)+L(2,4)	2+1=3	< 5	New route
5	C(1,2)+L(2,5)	2+Infinite=Infinite	=	No action
6	C(1,2)+L(2,6)	2+Infinite=Infinite	=	No action
7	C(1,2)+L(2,7)	2+4=6	6 < Infinite	New route

3. **Step 2**

The known route table now reads

Node	2*	3	4	5	6	7
Least cost path	2	**7**	**3**	Infinite	Infinite	**6**
Route	1-2	**1-2-3**	**1-2-4**			**1-2-7**

New and modified entries are shown **bold**.

Identification of least cost "unconsidered" node, identifies node 4. Examination of possible routes to other nodes proceeds thus.

Node	Formula	Value	Comparison	Result
2	C(1,4)+L(4,2)	3+1=4	4 > 2	No action
3	C(1,4)+L(4,3)	3+2=5	5 < 7	New route
5	C(1,4)+L(4,5)	3+2=5	5 < Infinite	New Route
6	C(1,4)+L(4,6)	3+Infinite=Infinite	=	No action
7	C(1,4)+L(4,7)	3+Infinite=Infinite	Infinite > 7	No action

4. **Step 3**

The known route table now reads

Node	**2***	**3**	**4***	**5**	**6**	**7**
Least cost path	2	**5**	3	**5**	Infinite	6
Route	1-2	**1-2-4-3**	1-2-4	**1-2-4-5**		1-2-7

New and modified entries are shown **bold**.

Identification of least cost "unconsidered" node, identifies node 5. Examination of possible routes to other nodes proceeds thus.

Node	Formula	Value	Comparison	Result
2	C(1,5)+L(5,2)	5+Infinite=Infinite	Infinite > 2	No action
3	C(1,5)+L(5,3)	5+Infinite=Infinite	Infinite > 7	No action
4	C(1,5)+L(5,4)	5+2=7	7 > 3	No action
6	C(1,5)+L(5,6)	5+1=6	6 < Infinite	New route
7	C(1,5)+L(5,7)	5+Infinite=Infinite	Infinite > 7	No action

5. **Step 4**

The known route table now reads

Node	**2***	**3**	**4***	**5***	**6**	**7**
Least cost path	2	5	3	5	**6**	6
Route	1-2	1-2-4-3	1-2-4	1-2-4-5	**1-2-4-5-6**	1-2-7

New and modified entries are shown **bold**.

Identification of least cost "unconsidered" node, identifies node 3. Examination of possible routes to other nodes proceeds thus.

Node	Formula	Value	Comparison	Result
2	C(1,3)+L(3,2)	5+5=10	10 > 2	No action
4	C(1,3)+L(3,4)	5+2=7	7 > 3	No action
5	C(1,3)+L(3,5)	5+Infinite=Infinite	Infinite > 5	No action
6	C(1,3)+L(3,6)	5+1=6	6 = 6	No action
7	C(1,3)+L(3,7)	5+1=6	6 = 6	No action

6. **Step 5**

   The known route table now reads

Node	**2***	**3***	**4***	**5***	**6**	**7**
Least cost path	2	5	3	5	6	6
Route	1-2	1-2-4-3	1-2-4	1-2-4-5	1-2-4-5-6	1-2-7

   Identification of least cost "unconsidered" node, identifies node 6. Examination of possible routes to other nodes proceeds thus.

Node	**Formula**	**Value**	**Comparison**	**Result**
2	C(1,6)+L(6,2)	6+Infinite=Infinite	Infinite > 2	No action
3	C(1,6)+L(6,3)	6+1=7	7 > 5	No action
4	C(1,6)+L(6,4)	6+Infinite=Infinite	Infinite > 3	No action
5	C(1,6)+L(6,5)	6+1=7	7 > 5	No action
7	C(1,6)+L(6,7)	6+3=9	9 > 6	No action

7. **Step 6**

   The known route table now reads

Node	**2***	**3***	**4***	**5***	**6***	**7**
Least cost path	2	5	3	5	6	6
Route	1-2	1-2-4-3	1-2-4	1-2-4-5	1-2-4-5-6	1-2-7

   Identification of least cost "unconsidered" node, identifies node 7. Examination of possible routes to other nodes proceeds thus.

Node	**Formula**	**Value**	**Comparison**	**Result**
2	C(1,7)+L(7,2)	6+4=10	10 > 2	No action
3	C(1,7)+L(7,3)	6+1=7	7 > 5	No action
4	C(1,7)+L(7,4)	6+Infinite=Infinite	Infinite > 3	No action
5	C(1,7)+L(7,5)	6+Infinite=Infinite	Infinite > 5	No action
6	C(1,7)+L(7,6)	6+3=9	9 > 6	No action

   All nodes have now been examined and the final routing table from node 1 is now.

Node	**2**	**3**	**4**	**5**	**6**	**7**
Least cost path	2	5	3	5	6	6
Route	1-2	1-2-4-3	1-2-4	1-2-4-5	1-2-4-5-6	1-2-7

4.2.4 Open Shortest Path First Protocol (OSPF)

One of the most widely used link-state routing protocols is OSPF.

The first word, “Open,” refers to the fact that it is an open, nonproprietary standard, created undethe supports of the IETF.

The “SPF” part comes from an alternative name for link state routing. OSPF adds quite a number of features to the basic link-state algorithm described above, including the following:

- **Authentication of routing messages:** Any host that has not been configured correctly may decide that it can reach every host in the universe at a cost of 0. When that host advertises this fact, every router in the surrounding neighborhood updates its forwarding tables to point to that host, and as a result, this host receives a vast amount of data, which becomes highly unmanageable. As a result, it typically drops the whole data and the network comes to a halt. Such disasters can be averted in many cases by requiring routing updates to be authenticated.

  Early versions of OSPF used a simple 8-byte password for authentication. This is not a strong enough form of authentication to prevent dedicated malicious users, but it alleviates many problems caused by misconfiguration.
- Additional hierarchy: Hierarchy is one of the fundamental tools used to make systems more scalable. OSPF introduces another layer of hierarchy into routing by allowing a domain to be partitioned into areas. This means that a router within a domain does not necessarily need to know how to reach every network within that domain—it may be sufficient for it to know only how to get to the right area. Thus, there is a reduction in the amount of information that must be transmitted to and stored in each node.

Each router has only the information about the routers in its own region and has no information about routers in other regions. So, routers just save one record in their table for every other region. In this example, we have classified our network into five regions (see below).

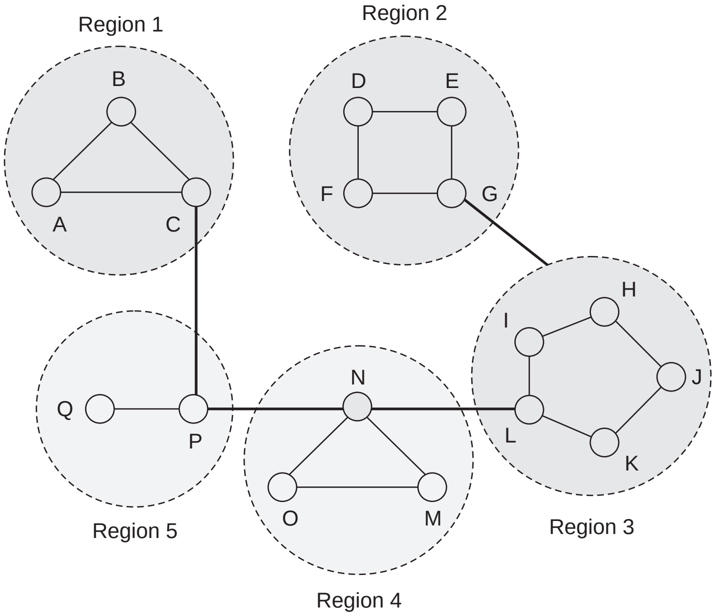


Figure 4.14: Classification of network into regions

Table 4.6: Routing table for OSPF

Destination	Line	Weight
A	---	---
B	B	1
C	C	1
Region 2	B	2
Region 3	C	2
Region 4	C	3
Region 5	C	4

If A wants to send packets to any router in region 2 (D, E, F or G), it sends them to B, and so on. As you can see, in this type of routing, the tables can be summarized, so network efficiency improves.

- Load balancing: OSPF allows multiple routes to the same place to be assigned the same cost and will cause traffic to be distributed evenly over those routes.

There are several different types of OSPF messages, but all begin with the same header, as shown in Figure below.

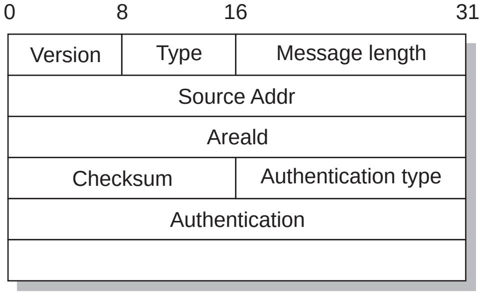


Figure 4.15(a): OSPF header format

The Version field is currently set to 2

Type field may take the values 1 through 5.

The SourceAddr identifies the sender of the message.

AreaId is a 32-bit identifier of the area in which the node is located.

The entire packet, except the authentication data, is protected by a 16-bit checksum using the same algorithm as the IP header.

The Authentication type is 0 if no authentication is used; otherwise it may be 1, implying a simple password is used, or 2, which indicates that a cryptographic authentication checksum.

We now see the five types of OSPF messages.

Type 1 is the "hello" message, which a router sends to its peers to notify them that it is still alive and connected as described above.

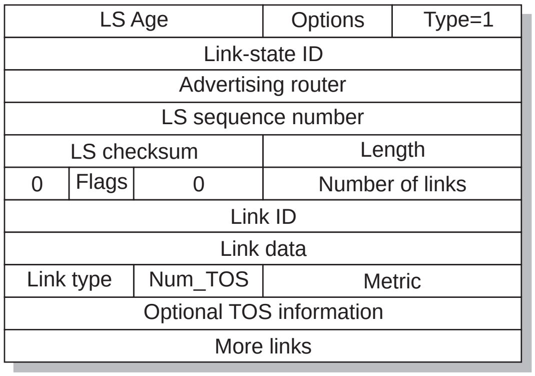
The remaining types are used to request, send, and acknowledge the receipt of link state messages.

The basic building block of link-state messages in OSPF is known as the link-state advertisement (LSA).

One message may contain many LSAs. We provide a few details of the LSA here. Like any internetwork routing protocol, OSPF must provide information about how to reach networks.

Thus, OSPF must provide a little more information than the simple graph-based protocol described above. Specifically, a router running OSPF may generate link-state packets that advertise one or more of the networks that are directly connected to that router. In addition, a router that is connected to another router by some link must advertise the cost of reaching that router over the link.

These two types of advertisements are necessary to enable all the routers in a domain to determine the cost of reaching all networks in that domain and the appropriate next hop for each network.



LS Age		Options	Type=1
Link-state ID			
Advertising router			
LS sequence number			
LS checksum		Length	
0	Flags	0	Number of links
Link ID			
Link data			
Link type	Num_TOS	Metric	
Optional TOS information			
More links			

Figure 4.15(b): OSPF link-state advertisement

The above Figure shows the packet format for a "type 1" link-state advertisement. Type 1 LSAs advertise the cost of links between routers. Type 2 LSAs are used to advertise networks to which the advertising router is connected, while other types are used to support additional hierarchy.

The LS Age is the equivalent of a *time to live*, except that it counts up and the LSA expires when the age reaches a defined maximum value.

The Type field tells us that this is a type 1 LSA. In a type 1 LSA, the Link-state ID and the Advertising router field are identical. Each carries a 32-bit identifier for the router that created this LSA. While a number of assignment strategies may be used to assign this ID, it is essential that it be unique in the routing domain and that a given router consistently uses the same router ID.

One way to pick a router ID that meets these requirements would be to pick the lowest IP address among all the IP addresses assigned to that router.

The LS sequence number is used exactly as described above, to detect old or duplicate LSAs. The LS checksum is similar to other protocols; it is of course used to verify that data has not been corrupted.

It covers all fields in the packet except LS Age, so that it is not necessary to re-compute a checksum every time LS Age is incremented.

Length is the length in bytes of the complete LSA. Now we get to the actual link-state information.

Each link in the LSA is represented by a Link ID, some Link Data, and a metric.

The first two of these fields identify the link; a common way to do this would be to use the router ID of the router at the far end of the link as the Link ID, and then use the Link Data to disambiguate among multiple parallel links if necessary.

The metric is of course the cost of the link. The Type tells provides some idea about the link, such as whether it is a point-to-point link or virtual circuit.

The TOS information is present to allow OSPF to choose different routes for IP packets based on the value in their TOS field. Instead of assigning a single metric to a link, it is possible to assign different metrics depending on the TOS value of the data.

For example, if we had a link in our network that was very good for delay-sensitive traffic, we could give it a low metric for the TOS value representing low delay and a high metric for everything else. OSPF would then pick a different shortest path for those packets that had their TOS field set to that value. It is worth noting that, at the time of writing, this capability has not been widely deployed.

4.2.5 Address Resolution Protocol (ARP)

Overview

Address Resolution Protocol (ARP) hovers in the shadows of most networks. The address resolution protocol (ARP) maps IP network addresses to the hardware (or MAC) addresses that are used by a data link protocol. This protocol is mostly used when IPv4 is used over the Ethernet and it functions below the network layer as an interface in between the network and link layer.

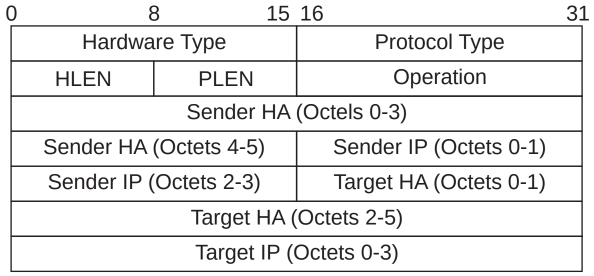
Address resolution would mean discovering the address of a host in a network. An Ethernet network makes use of hardware addresses to specify the sender and receiver of every frame being sent over the network. The hardware or physical address is also known as the Medium Access Control (MAC) address, with respect to Ethernet standards. Network Interface cards are assigned unique 6 byte link addresses by the manufacturer and are stored in the PROM.

Now, the Ethernet address in question is actually the link layer address and is based on the interface card used. On the other hand, IP, the protocol for the network layer above the link layer, does not care about *link addresses* of the source or destination. Therefore, ARP is needed to map these two types of addresses: MAC addresses and IP addresses. The *arp* client and server processes are present on all computers that use IP over Ethernet.

The *arp* messages used in the ARP protocol are of four types: ARP request, ARP reply, RARP request, and RARP reply.

The format of an *arp* message which is used to resolve a MAC address is shown below:

Hosts dynamically obtain the MAC address corresponding to an IP network-layer address by broadcasting ARPs. On receiving the MAC addresses, the IP device normally caches the just-acquired IP-to-MAC address mapping, thus reducing the need to send many address resolution requests. If the device does not respond within a stipulated time, the cache entry is flushed. This is a good practice as the arp cache is of a finite size and would be full in no time if it was filled up with incomplete or obsolete entries.



0	8 15	16 31
Hardware Type		Protocol Type
HLEN	PLEN	Operation
Sender HA (Octels 0-3)		
Sender HA (Octets 4-5)		Sender IP (Octets 0-1)
Sender IP (Octets 2-3)		Target HA (Octets 0-1)
Target HA (Octets 2-5)		
Target IP (Octets 0-3)		

Figure 4.16: Format of an ARP message

There is also the ***Reverse Address Resolution Protocol*** (RARP), which does the reverse process: maps MAC addresses to their corresponding IP addresses. The diskless computers that have no idea of their IP address when booted could use this protocol. RARP relies on the presence of a RARP server with table entries of MAC-layer-to-IP address mappings.

4.3 INTERNET CONTROL MESSAGE PROTOCOL (ICMP)

The Internet Control Message Protocol (ICMP) is a network-layer internet protocol that compensates for the lack of error control and assistance mechanisms in Internet Protocol (IP). In short, it is a companion to IP. ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route.

The Internet Protocol is not designed to be absolutely reliable. The purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. There are still no guarantees that a datagram will be delivered or a control message will be returned. Some datagrams may still be undelivered without any report of their loss.

The higher level protocols that use IP must implement their own reliability procedures if reliable communication is required. The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages.

IP implementations are required to support this protocol. ICMP is considered an integral part of IP, although it is architecturally layered upon IP. ICMP provides error reporting, flow control and first-hop gateway redirection.

As you already know, IP does not provide reliable delivery of data. It is a connectionless protocol and has no means to report errors or even correct any errors that might occur during the transmission of the datagram. Most of the time datagrams are discarded by routers because it couldn't find a router to the final destination. Other situations that could occur are the TTL (time to live) field becoming zero, or not all fragments of a datagram reaching within the time limit. Since IP does not have any means to report these errors or even correct them, another protocol was required. This protocol should also be able to query hosts and routers as IP could not even determine if a particular host or router was even alive. Thus ICMP was designed to solve these issues.

The ICMP server does the following functions:

- It is executed on all IP end systems (i.e. computers) and IP intermediate systems (i.e. routers)
- Responsible for reporting any issue with the delivery of IP datagrams in the IP network

- Shows if :
  - o An end system is not responding
  - o An IP network is not reachable
  - o If a node is overloaded
  - o If there is an error in the IP header
- Network administrators or internet managers use ICMP to check if routers are directing packets to the correct destination.

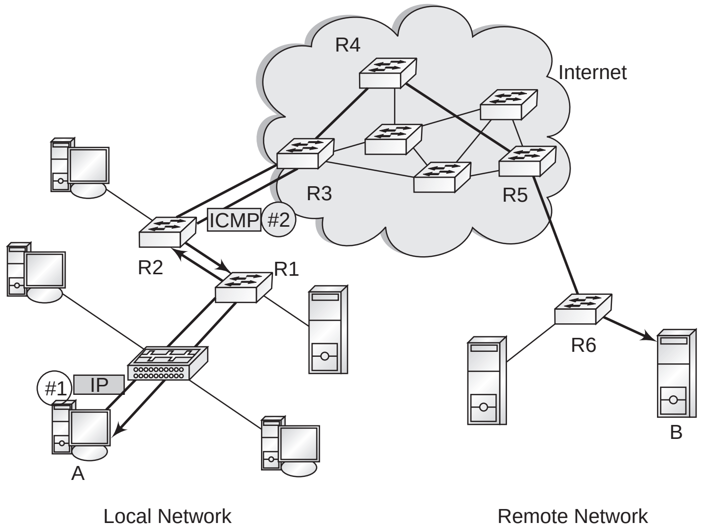


Figure 4.17(a): Working of ICMP

The diagram shown gives a typical example of how ICMP works. Host A is trying to send an IP datagram to a host B on another network. In the course of its transmission, it encounters a problem at a router (R3) because of which the datagram had to be dropped. In such situations, the router R3 would send an ICMP message back to the source host A letting it know that a problem occurred. The message will contain error information as well which could help host A solve the problem. Notice how the ICMP message has to go back all the way to host A, and not just to the routers on the way (R2 or R1).

4.3.1 ICMP Messages

The ICMP messages are organized into two broad categories:

- **Error-reporting messages:** These messages provide feedback to the sender if a router or a host encounters errors when a packet is processed.
- **Query Messages:** These messages do not indicate errors, instead, it helps obtain information from routers or hosts. They mostly work in pairs (as request/reply pairs) and can implement testing, IP-related features, let hosts learn about routers within the network, and even help routers route their messages.

Message Format

The ICMP message format has an 8 byte header and the data part is variable-sized. The header can vary for the different message types (will be discussed later) but the beginning four bytes are common to all ICMP messages.

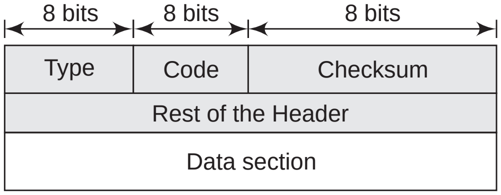


Figure 4.17(b): Format of an ICMP message

The various fields are:

- **ICMP type (8 bits):** There are different types of messages in ICMP and each of them has their own unique *Type* value. Since this field is 8 bits long, 256 message types can be specified, theoretically. ICMPv4 and ICMPv6 have different *Type* values.
- **ICMP code (8 bits):** Defines the exact purpose for the particular message type. In a way, this field can be known as the message subtype. Again, 256 codes or subtypes can be there for each ICMP message type.
- **Checksum**
- **Rest of the header:** Is specific for each message type.

Some of the ICMP messages with their type values are given below:

Table 4.7: ICMP message types and their values

Message Class	Type Value	Message Name
ICMPv4 Error messages	3	Destination Unreachable
	4	Source Quench
	5	Redirect
	11	Time Exceeded
	12	Parameter Problem
ICMPv4 Query Messages	0	Echo Reply
	8	Echo (Request)
	9	Router Advertisement
	10	Router Solicitation
	13	Timestamp (request)
	14	Timestamp reply
	15	Information Request
	16	Information Reply
	17	Address Mask Request
	18	Address Mask reply
	30	Trace route

ICMPv6 Error messages	1	Destination Unreachable
	2	Packet Too Big
	3	Time Exceeded
	4	Parameter Problem
ICMPv6 Query messages	128	Echo Request
	129	Echo Reply
	133	Router Solicitation
	134	Router Advertisement
	135	Neighbor Solicitation
	136	Neighbor Advertisement
	137	Redirect
	138	Router Renumbering

The *destination unreachable* message is usually sent by the router when it cannot send the packet to its destination. The packet will then be discarded by the router. There are four destination-unreachable messages: *network unreachable* (the packet could not be routed or the addressing is wrong), *host unreachable* (could mean that the subnet mask is wrong thus the packet could not be delivered), *protocol unreachable* (occurs when the receiver is not compatible with the upper layer protocol mentioned in the packet), *and port unreachable* (Non-availability of the TCP socket or port).

In the query messages, the *echo request* message (sent using the ping command) checks if a node can be reached in a network. The *echo reply* message would be generated if the node is reachable.

The *Redirect* message encourages the router to do a more efficient routing. The *Time exceeded* ICMP message will be sent by a router if the packet's TTL field has reached a value of zero. After sending this message, the router will get rid of the packet.

4.3.2 ICMP Router-Discovery Protocol (IDRP)

The ICMP router discovery protocol (IRDP) allows host computers to discover the router IP address on directly attached subnets and thus use it as the default gateway. This protocol thus removes any need to manually configure router addresses. IRDP is not dependent on any particular routing protocol.

IDRP makes use of ICMP *router advertisements* and *router solicitation* messages to determine the IP addresses of routers on the network. Operational routers keep multicasting *router advertisement* messages at regular intervals from each of its interfaces, thus announcing the router IP address. Host computers continuously listen for these messages. Host computers have the option of not waiting for the router advertisement and sending a multicast router solicitation message to get the router addresses.

Though the router discovery messages help host computers in their search for neighboring routers, it does not help discover the best router to reach a destination.

4.4 IPv6

IPv4 is still widely used as the internet protocol but experts believe that the four billion addresses that could be allocated using the 32-bit IPv4 address format are all but exhausted now. This has happened because government agencies are being allocated multiple large blocks of addresses. Address-depletion is a major concern for the Internet. Thus, IPv6 came into the scene with an expansion in addressing from 32-bit (4octets) addresses to 128-bit (16 octets) addresses.

This could be the answer to all the requests for more addresses, but it has not become a standard as yet. The IPv4 protocol could not support real-time audio and video transmissions, and encryption and authentication of data in some applications as well. Another much advanced protocol was overdue. Other interesting facts about IPv6 are the better unicast and broadcasting methods, usage of hexadecimal numbers in the IP address format, and the elimination of decimal dots with the introduction of colons (":") as delimiters in the address.

4.4.1 Description of IPv6 Packet Header

The IPv6 packet contains the header and the payload. The header part contains the information required for the packet to reach its destination. The payload is the actual data. The IPv6 header is surprisingly simpler than the IPv4 header.

To understand the IPv6 header, let us see a pictorial comparison of the IPv4 and IPv6 headers.

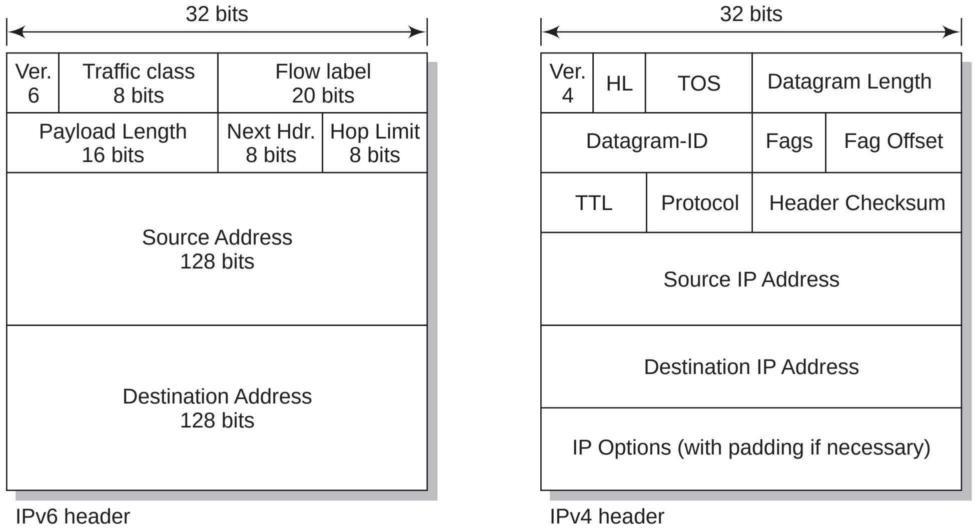


Figure 4.18: Comparison of IPv6 and IPv4 header formats

The header is of length ***40 bytes*** and has eight fields. Some of these fields could be seen in the IPv4 header as well. The various fields are:

1. **Version:** Informs the routers which Internet Protocol version is in use. It would be 6 for IPv6.
2. **Traffic Class (8 bits):** This field is similar to the ***Type of service*** field in IPv4. It gives the class of service (CoS) priority of the packet. Using this field, devices can distinguish between latency sensitive traffic (voice, video etc) and low-priority data (email, web traffic etc).

3. **Flow Label (20 bits):** The host uses this field so that special handling is obtained from the routers that are IPv6-compliant. It indicates which packets are a part of a special flow or the ones that need a special class of service (CoS). This management of traffic flow between stations helps in giving a quality of service.
4. **Payload Length (16 bits):** Describes the length, in octets, of the IPv6 payload (data) part of the IPv6 packet. The 16-bit field length (2^{16}) lets version 6 support payloads in excess of 64,000 octets.
5. **Next Header (8 bits):** This field was initially the protocol field in IPv4. It alerts routers about the next extension header that should be examined.
6. **Hop limit (8 bits):** Specifies the number of routing hops a packet is allowed to take before being discarded. The field is 8 bits in length, hence the maximum hops a packet can take is 255. No path would be that long in today's networks.
7. **Source Address (128 bits):** Gives the 128 bit address of the source node.
8. **Destination Address (128 bits):** Specifies the 128-bit address of the final destination node.

Extension Headers

There is an option of placing extension headers in between the IPv6 header and the upper-layer header of the packet for optional Internet layer information like source routing, encryption, and authentication. The extension headers, if more than one, can be held together by using the ***Next Header*** field in the IPv6 header. It informs the router which extension header should be next in line.

If there are no extension headers, the ***Next header*** field would indicate the upper layer header like the TCP header, UDP header, etc.

4.4.2 Addressing Description

The IPv6 address is 128 bits long, or 16 bytes (octets). The address is specified in a ***hexadecimal colon notation.*** Hexadecimal numbers are base 16. The following figure describes the how the IPv6 address is represented. The 128 bits are divided into eight parts, with each part being 2 bytes (16 bits) in length. Converting 2 bytes to the hexadecimal format would become four hexadecimal digits. This means that the 128 bit address in a hexadecimal format would have 32 hexadecimal digits. Notice the colon after every four hexadecimal digits.

Therefore, each part *x* would be a 16-bit binary number and there are eight parts in total. This would mean that the total length would be equal to 16 * 8 = 128 bits.

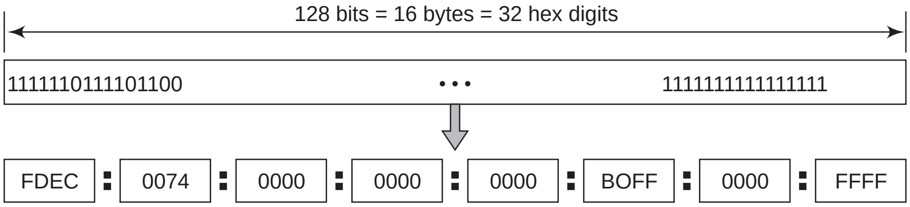


Figure 4.19(a): IPv6 address in hexadecimal colon notation

Abbreviated IPv6 address

Even after conversion to the hexadecimal format, the IPv6 address is really long. But there are many cases where the digits are zeroes, and then the address can be abbreviated or shortened. The ***leading zeroes*** can be left out thereby shortening the IP address. Remember that no trailing zero can be dropped.

Numbers like 0005 can be reduced to just 5 and 0000 as 0. But numbers like 5500 cannot be shortened. If there are consecutive parts consisting of zeroes only, then the zeroes can be removed and replaced by just a semicolon. But this is a one-time only possibility in an address. This means that even if there are two parts of only zeroes in an address, only one of them can be shortened to a semi-colon. The following diagram gives an example of an abbreviated IPv6 address.

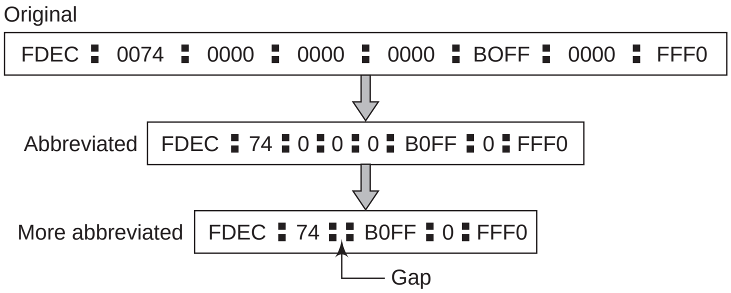

Original
FDEC : 0074 : 0000 : 0000 : 0000 : BOFF : 0000 : FFF0
Abbreviated
FDEC : 74 : 0 : 0 : 0 : B0FF : 0 : FFF0
More abbreviated
FDEC : 74 :: B0FF : 0 : FFF0
Gap

Figure 4.19(b): Abbreviated IPv6 address

The question arises as to how can one get the original address from the abbreviated address? This can be understood by an example:

Abbreviated IPv6 address: **1:12:: 5:10:12B0**

Now, align the left part of the double colon to the left hand side of an original hexadecimal format of the IPv6 address and the right side of the double colon to the right side of the original format to understand how many parts need to be replaced by zeroes.

Example of an IPv6 address: xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx

1: 12: : 5: 10:12B0

Original address would be: 00 Separated double colon 0005:0010:1280

4.4.3 Broadcasting Methods

Since the IPv6 address is 128 bits long, there are **2^{128}** addresses available to be assigned to hosts on the Internet. The address space thus becomes much larger than what was available for the IPv4 addresses.

The IPv6 address has been categorized into several types with some of its leftmost bits being the defining factor. These leftmost bits, referred to as the ***type prefix*** is not fixed in length but ensures that no code is similar to the beginning part of any other code. This eliminates any uncertainty that could occur as one can just look at an address and determine its type prefix. The Table 4.8 shows the type prefixes with the type of address.

Table 4.8: Type prefixes for address types

Type Prefix	Type
0000 0000	Reserved
0000 0001	Unassigned
0000 001	ISO network addresses
0000 010	IPX (Novell) network addresses
0000 011	Unassigned
0000 1	Unassigned
0001	Reserved

001	Reserved
010	**Provider-based unicast addresses**
011	Unassigned
100	Geographic based unicast addresses
101	Unassigned
110	Unassigned
1110	Unassigned
1111 0	Unassigned
1111 10	Unassigned
1111 110	Unassigned
1111 1110 0	Unassigned
1111 1110 10	Link Local addresses
1111 1110 11	Site local addresses
1111 1111	Multicast addresses

IPv6 includes some new broadcasting methods:

1. Unicast
2. Multicast
3. Anycast

In the case of IPv4 addresses, it was never understandable whether an address was assigned to a node or a network interface. When it came to IPv6, designers made sure that an address was defined or configured on each network interface.

1. Unicast Addresses

Any packet being sent to a unicast address will be received by a specific computer. As the name suggests, a unicast address is the address of one computer. There can be two types of unicast addresses: provider-based and geographically based. Since the geographically based unicast addresses are reserved for future use, we will discuss provider-based unicast addresses.

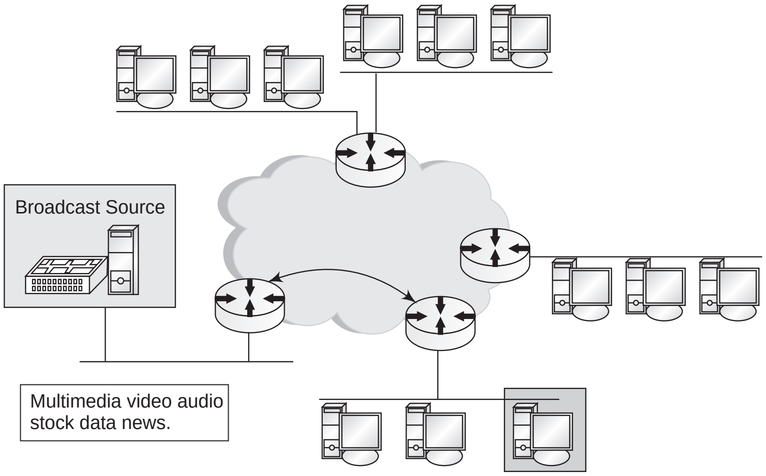


Figure 4.20: Broadcasting over unicast addresses

The following figure shows the prefixes for a provider-based unicast address.

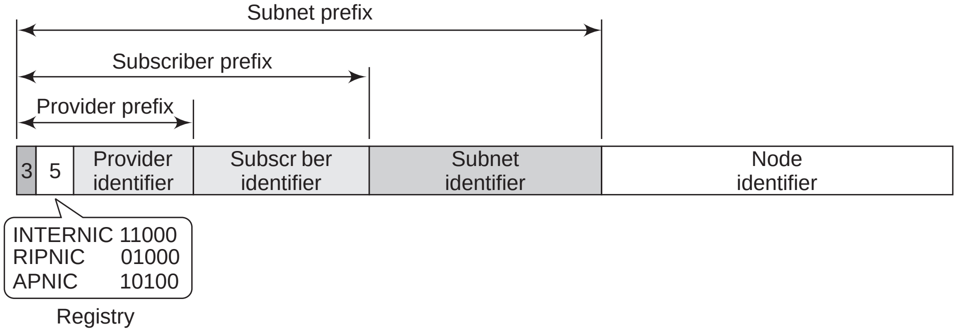


Figure 4.21: Prefix for provider-based unicast addresses

1. **Type identifier (3 bits):** The first three bits specify that the address is provider-based. Referring to the previous table which showed the type prefixes, these three bits would be 010.
2. **Registry Identifier (5 bits):** Identifies the registry center that has registered the address.

RegistryCenter	**Code**
INTERNIC (NorthAmericanCenter)	**11000**
RIPNIC (Europe)	**01000**
APNIC (Asia-Pacific)	**10100**

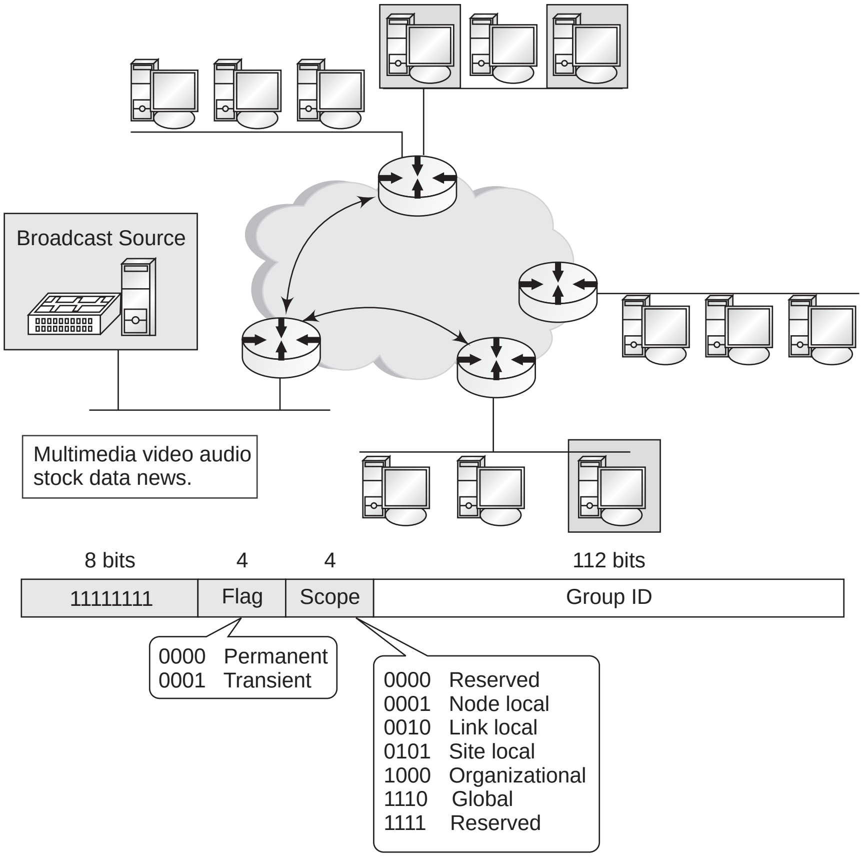
3. **Provider Identifier (Variable length, recommended-16 bits):** Specifies the Internet access provider.
4. **Subscriber identifier (recommended-24 bits):** Identifier given to the organization that subscribes to the Internet via a provider.
5. **Subnet identifier (recommended-32 bits):** Specifies the sub-network in the subscriber's territory.
6. **Node Identifier (recommended- 48 bits):** Identifies the node in the subnet. The node identifier may *equate* the length of the node's physical address in the future.

2. Multicast Addresses

A multicast address is used to send packets to a group of computers. This means that multiple hosts may be registered to a multicast group, and this group has a multicast address assigned to it. Thus, by specifying just the multicast address, the sending host can send a packet to all hosts in that particular multicast group.

The multicast address format is shown in the following diagram. As specified in the type prefix table, multicast addresses will have the type prefix as 1111 1111.

Flag (4 bits): Indicates if the group address is permanent (0000) or transient (0001). Transient group addresses are temporary addresses whereas permanent group addresses are accessible all the time. They are specified by the Internet authorities.



8 bits	4	4	112 bits
11111111	Flag	Scope	Group ID

0000 Permanent
0001 Transient

0000 Reserved
0001 Node local
0010 Link local
0101 Site local
1000 Organizational
1110 Global
1111 Reserved

Figure 4.22: Broatcasting over multicast addresses

Scope (4 bits): Defines the scope of the group address.

3. Anycast Addresses

The *anycast* address defines a set of hosts, just like the multicast address, but when packets are sent, it is delivered to the nearest interface defined by that address.

So, in contrast to transmitting the packet to all nodes in a group as in the case of multicast, anycast would only send the packet to the nearest node in the anycast group.

SUMMARY

- The Internet Protocol Suite consists of a set of communication protocols that the Internet and many other commercial networks run on.
- This protocol suite is also referred to as the ***TCP/IP protocol suite*** after its two best-known communication protocols — Transmission Control Protocol (TCP) and Internet Protocol (IP).
- The ***Internet Protocol (IP)*** is the primary network layer protocol used for data communication across packet-switched internetworks.
- An ***IP address*** is the term given for a logical address in the network layer of the TCP/IP protocol suite. It has a basic format made up of components divided by decimal dots.

- Every IP address consists of two parts, one identifying the network that the computer belongs to known as the ***Network ID or Network Address*** and one identifying the host on the network known as the ***Host ID or Host Address.***
- The address resolution protocol (ARP) maps IP network addresses to the hardware (or MAC) addresses that are used by a data link protocol.
- IP routing refers to the set of protocols that decide which path data packets take when traveling from their source towards their destination.
- The Internet Control Message Protocol (ICMP) is a network-layer internet protocol that compensates for the lack of error control and assistance mechanisms in Internet Protocol (IP).
- The ICMP router discovery protocol (IRDP) allows host computers to discover the router IP address on directly attached subnets and thus use it as the default gateway.
- IPv6 expanded the addressing from 32-bit (4octets) addresses to 128-bit (16 octets) addresses.
- The IPv6 address is 128 bits long or 16 bytes (octets) and is specified in a ***hexadecimal colon notation.***
- IPv6 includes some new broadcasting methods: **Unicast, Multicast, and Anycast.**

5

Transport Process

Introduction

The transport layer, the fourth layer in the OSI reference model, is responsible for process-to-process communication and delivery of a message. Processes refer to the application programs running on the computer. It is the transport layer that understands whether the packets being sent belong to one message or not. It ensures that the message arrives at the destination in order and safe. The network layer, on the other hand, treats each packet as a separate entity and does not understand the relationship between the packets.

Transport protocols can be of two types: connectionless and connection-oriented. The connectionless protocol, like UDP, treats each segment as an independent packet. Connection-oriented protocols like TCP, establishes a connection with the transport layer at the destination and then it begins to transfer the packets. The connection is terminated when data transfer is over. In addition, transport protocols take care of flow and error control as well. UDP does not implement any flow or error control mechanisms but TCP does support flow control mechanisms like the sliding window.

5.1 THE TRANSPORT SERVICE

The Transport Layer in the TCP/IP protocol model provides the mechanism for process-to-process communication. It utilizes the services of the Network layer below it in the movement of data from the sender to the destination. When comparing the functions of the network layer and the transport layer, the network layer provides the service of data transfer between end systems while the transport layer transfers data between processes. The Transport Layer is solely responsible for providing logical communication (logical end-to-end transport) between application's processes that are running on different hosts.

The protocols defined for the transport layer will primarily run on end systems. Two protocols — UDP and TCP have been defined for this layer.

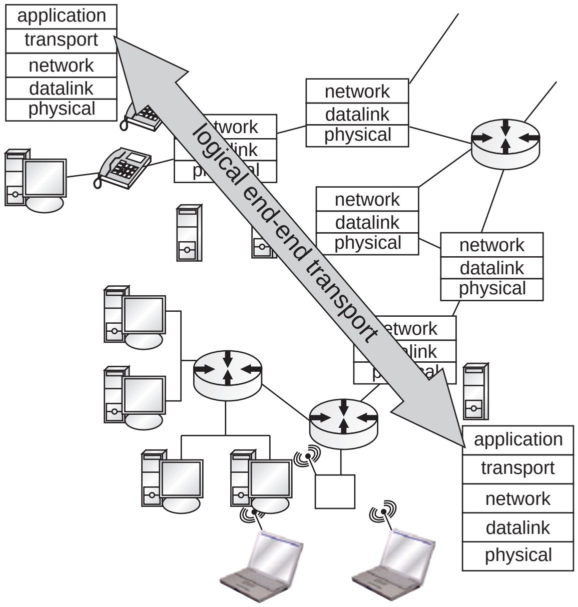


Figure 5.1: Data transfer across transport layer

UDP, the connection-less protocol, does not provide reliable data transfer but its simplicity effects an efficient data transfer. Hence, applications that do not pay attention to reliability and can tolerate loss of data to some extent but have efficiency as the most important criteria can use this protocol.

TCP, the transport control protocol, offers a reliable and stream-oriented data transfer service. The TCP/IP model and its popularity are all due to TCP and its process-to-process communication mechanism. This protocol implements error control and flow control in the data stream ensuring the reliability of data being sent as well.

Using unique numbers known as port numbers and the technique of multiplexing/de-multiplexing, several processes can use UDP and TCP simultaneously.

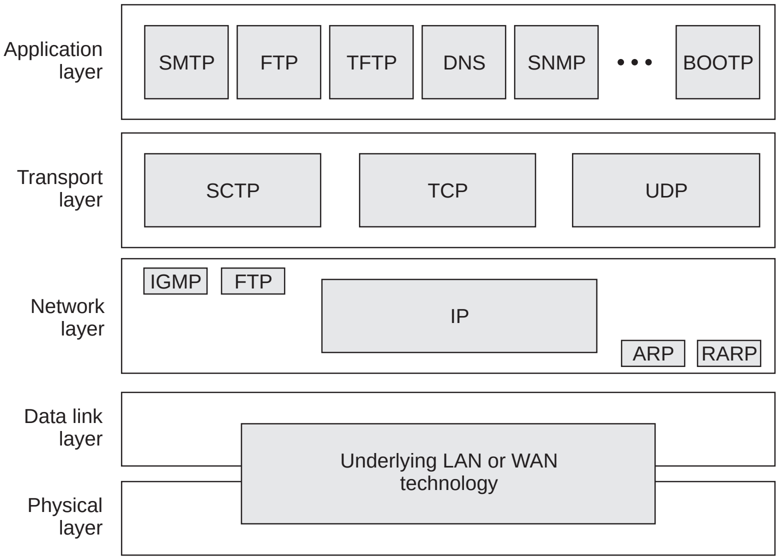


Figure 5.2: Layer-wise protocols used

Transport protocols utilize network protocols like IP for the datagram delivery. Data delivery over the Internet could get lost, delayed, or even duplicated. Transport protocols like TCP compensate for these problems by having flow and error control mechanisms in place.

Note the various protocols at the different layers in the OSI model in the figure shown. Transport protocols that are widely used are TCP, UDP, and SPX. They utilize the services of network layer protocols like IP, ICMP, and IPX.

Of these protocols, some are connection oriented and some are connection-less. If a protocol meets one of the following criteria, it is categorized as a connection-oriented protocol:

- A virtual path is established between two nodes and data is transferred over it
- Or, the protocol has an error –recovery process.

Examples of connection oriented protocols are Frame Relay, TCP, SPX, and X.25. Some of the connectionless protocols are IP, IPX, and UDP.

The transport layer protocols that have error recovery implemented (e.g. TCP) will be connection oriented,

- Connection Oriented
- The datagrams will have fields in the header which allows receivers to acknowledge its receipt and also for error-checking (e.g. checksum)
- The sending host demands acknowledgement of packets that have been received safe and sound.

5.2 FUNCTIONS OF TRANSPORT LAYER

The transport layer performs mainly four functions:

- **Segmenting and assembling upper-layer applications:** At the sender side, the data that is coming from the upper application layer, is segmented into datagrams with a transport protocol (UDP or TCP) header and then passed on to the lower network layer. At the destination, the transport layer does the opposite of removing the header and assembling the segments and passing it to the application layer.
- **Transport of segments from host-to-host:** The primary function of the transport layer is its part in the communication or delivery of segments from the sending host to the receiving host.
- **Establishment and management of end-to-end operations:** Connection oriented transport protocols establish connections between the transport layers in both hosts after which, data is sent.
- **Error recovery:** The transport protocols, using a variety of error control mechanisms like checksum, acknowledgement etc, can detect if there are any errors that might have occurred during the transmission.

In addition, the Transport Layer does the following functions:

- Verifies the integrity of data sent via flow control techniques like sliding window mechanism.
- Multiplexes the data from the upper layer as more than one application program will want to use the transport protocol simultaneously.
- For communication across the network, it is the transport layer that sets up and tears down the virtual circuits created so that data can be delivered from host-to-host.

- Ensures that information that could create confusion to the users like network dependent data is hidden from the upper layers.
- Chops up the datagrams from the session layer (layer 5) into segments to be sent to the lower Network Layer (Layer 3).
- Responsible for monitoring and ensuring that data reaches its destination safe and sound, with no errors.
- Provides general connection management and data transfer services.
- Provides the service of reliable delivery of data.

5.3 USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is the ***connection-less, unreliable*** transport layer (Layer 4) protocol. It is an interface between the Internet Protocol (IP) and the upper layer processes. UDP does not add anything to the services of IP (the network layer protocol). It only provides a process-to-process communication. This protocol does not do much in error checking, and has been overtaken by many other alternate technologies.

In spite of all this, UDP has its advantages. It is a *lightweight* layer above the Internet Protocol with a small overhead and a very simple protocol. This protocol can be used in those situations where a small message has to be sent and reliability is not a deciding factor. The interaction between the sending host and the receiving host is less if the transport layer protocol is UDP when compared to other protocols like TCP or SCTP.

UDP adds no flow control, reliability, or error-recovery features to IP. This simplicity of the protocol ensures that the UDP header has lesser bytes and consumes lesser network overhead than TCP. Therefore it is faster in establishing a connection with the destination process. If reliability is not so important, then bulk transfer would be faster. Most multimedia applications need a bulk transfer of data with no requirement of *fool-proof* reliability. To such applications, the reliability offered by the underlying networks is sufficient enough. All these reasons make UDP a much-desired transport layer protocol. Several widely used application layer protocols like Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and Trivial File Transfer Protocol (TFTP) use UDP.

Application	e-mail, telnet
Transport	TCP and UDP
Network	IP
Link	datalink

IP is the protocol defining the network layer and is responsible for host-to-host communication. It delivers the message to the destination host system. However, this is not enough as the message as to reach the correct *process* in the destination. Transport protocols like UDP do exactly that: process-to-process communication.

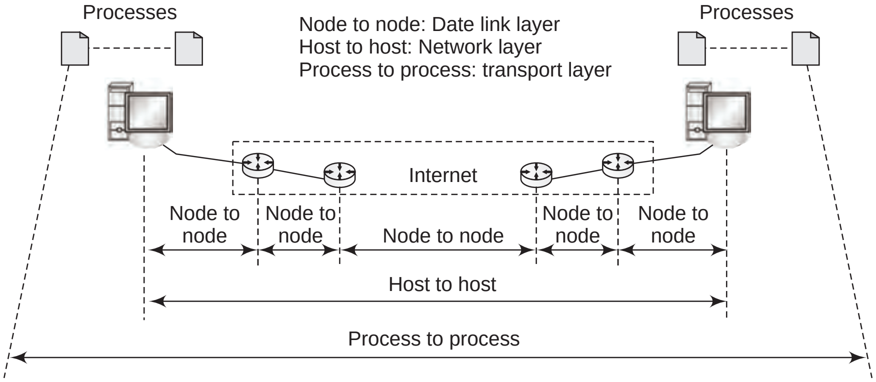


Figure 5.3: Process to process interaction using transport layer

5.3.1 Well-known Port Numbers

To uniquely identify the network application process running on the host machine that requires TCP/IP communication, a 16-bit *port number* is assigned to it. Computer networking has advanced so much that multi-port and multi-programming is very common. Computers can be local or remote and can have many server programs running simultaneously. The local host, the local process, the remote host, and the remote process all need unique identification numbers for communication.

The local and remote hosts are given IP addresses as a means of identification. The processes working on these hosts require the port numbers which can range from 0 – 65,535.

The client process/program is given a random port number by the UDP software running on the client host. This port number is referred to as the *ephemeral port number.*

The server process needs a port number as well but it cannot be a random one. This is because if the server process has a random number, then client processes who want to communicate with the server will not know which port number it should communicate with. It can request for the port number by sending a packet but this would result in unnecessary overhead. This can be avoided by using universal port numbers for servers known as ***well-known port numbers.***

By having the well-known port numbers, every client process will know the port number of any server process. Take the example of a client process (daytime) wanting to know the date and time. It will have an ephemeral port number of say, 25,000 as an identifier but the daytime server process will have the well-known port number of 13.

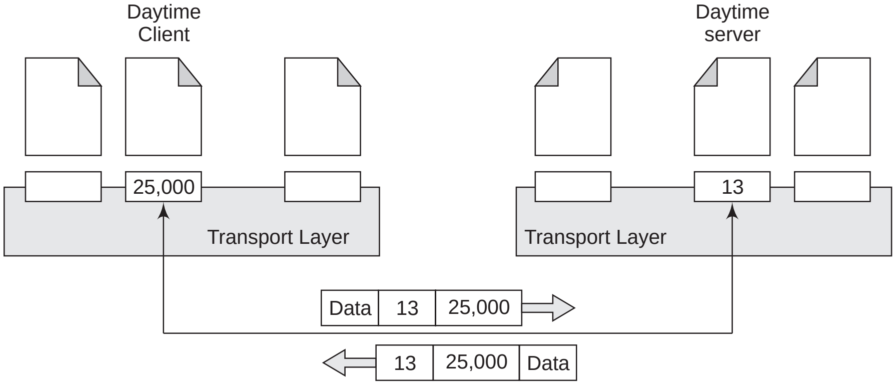


Figure 5.4(a): Example showing use of well-known port numbers

Now, the functions performed by IP addresses and port numbers can be understood by the following figure. Both together play the role of determining the final destination for the data. The IP address for the destination will identify which host is the destination amongst all the hosts in the network. Once the host has been decided upon, the port number will select the process.

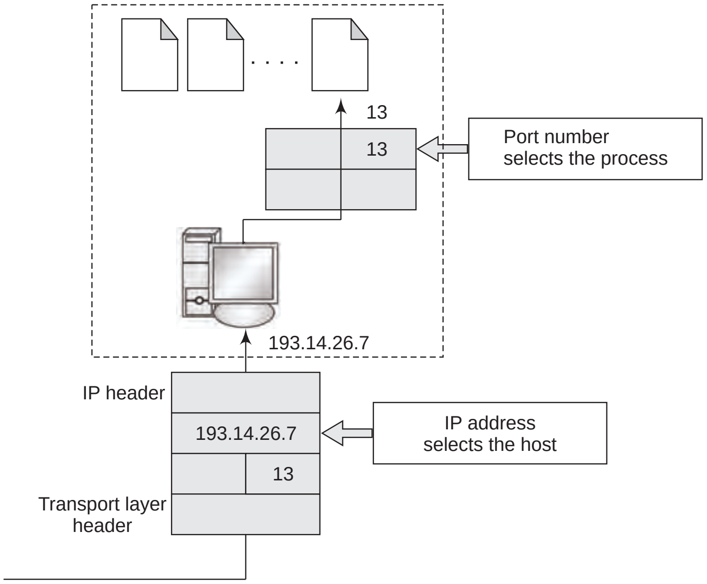
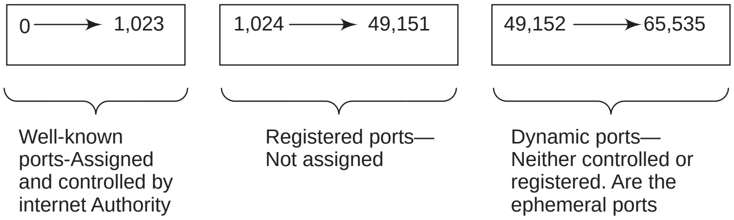


Figure 5.4(b): Functions performed by IP address and port numbers

The port numbers from 0-65,535 are divided into three ranges:


0 → 1,023
1,024 → 49,151
49,152 → 65,535
Well-known ports-Assigned and controlled by internet Authority
Registered ports— Not assigned
Dynamic ports— Neither controlled or registered. Are the ephemeral ports

Some of the well-known ports used by UDP are listed in the following table:

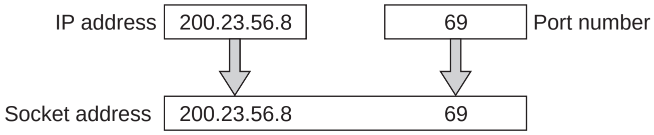
Table 5.1: Well-known UDP ports

Port No.	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
13	Daytime	Gives the date and time
17	Quote	Returns
20	FTP-DATA	File Transfer (Default data)
21	FTP	File Transfer (Control)
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer

37	TIME	Returns the time
53	Name Server	Given the domain name, it returns the IP address
69	TFTP	Trivial File Transfer
79	FINGER	Finger
110	POP3	Post Office Protocol v3
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
143	IMAP2	Interim Mail Access Protocol v2
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol – trap

5.3.2 Socket Addresses

The IP address and the port number together comprise the socket address. As mentioned, the IP address is an identifier for a host on the Internet and the port number identifies the process on the host. The socket address, a combination of these two, will uniquely identify a process on the Internet.



Therefore, in communication using UDP, a client socket address and a sever socket address is required. The client socket address will contain the client process's port number and IP address while the server socket address will have the server process's port number and IP address. The four-abovementioned information will be present in the IP and UDP header. The IP header will contain the IP addresses whereas the UDP header will contain the port numbers.

5.3.3 UDP Datagram

The UDP datagram consists of one single *unit* of binary data with the first eight bytes (fixed header size) being the header and the remaining being the data.

The UDP header consists of four fields each of two bytes:

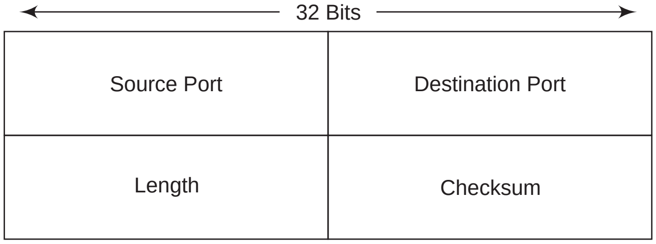


Figure 5.5: UDP header

1. **SourcePort number (16 bits/2 bytes):** Is the port number used by the sending application to send UDP datagrams. Since the length is 16 bits, the port number can be between 0-65,535.
2. **DestinationPort (16 bits):** Port number used by the recipient of the packet to accept the datagram.
3. **Length (16 bits):** Defines the total length of the datagram (header + data). Since the header is of fixed length, it would therefore indicate the length of the variable sized data part. Theoretically, the maximum size of the datagram would be 65,535 bytes. However, since the UDP user datagram has to be stored inside an IP datagram, the length has to be a smaller number, even as small as 8192 bytes.
4. **Checksum (16 bits):** This safety feature is used to identify if there are any errors in the datagram (header + data).It is the calculated value as a result of encoding of the data at the sender side. It is calculated by the receiver later on as well. If the datagram is damaged or tampered with during the data transmission, the value calculated by the sender and receiver will not be the same, by which the UDP protocol detects an error in the datagram. The calculation of the checksum and even including it in the datagram is optional for the UDP protocol. If it is not calculated, this field has only 1s.

5.3.4 UDP Checksum and its Computation

IP and other lower layer protocols do have the provision of a checksum but it detects errors only in the header. The UDP checksum checks for errors in the entire datagram, which means, both header and data are checked for errors.

The checksum calculation in UDP is different from the calculation seen in IP and ICMP. The UDP checksum includes three parts:

1. **Pseudo-header:** This header is part of the IP packet header. The datagram is encapsulated in this pseudo-header with some of its fields containing only 0s.
2. **UDP header:** This is the UDP datagram header.
3. **Data:** The data that is coming from the upper (application) layer.

The pseudo-header is a derivative of the IP header that has the destination IP address, source IP address, and protocol numbers. If there is no pseudo-header, the user datagram *may* reach its destination safely.

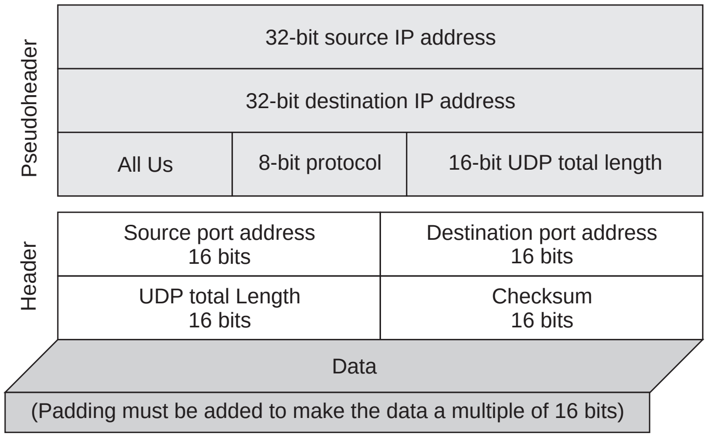

Pseudoheader
32-bit source IP address
32-bit destination IP address
All Us
8-bit protocol
16-bit UDP total length
Header
Source port address 16 bits
Destination port address 16 bits
UDP total Length 16 bits
Checksum 16 bits
Data
(Padding must be added to make the data a multiple of 16 bits)

Figure 5.6(a): UDP checksum

But if the IP header has got corrupted in the course of transmission, the packet will reach the wrong host. Therefore, the pseudo-header ensures that the user datagram reaches the required process on the intended host and via the intended transport protocol only. The pseudo-header in combination with the UDP header uniquely identifies the destination process.

The protocol field in the pseudo-header ensures that the packet belongs to UDP and not any other protocol. For UDP, the protocol field = 17. Say, this value changes during the transmission, then when the checksum is calculated at the receiver end, it detects the change in value and UDP ends up dropping the packet.

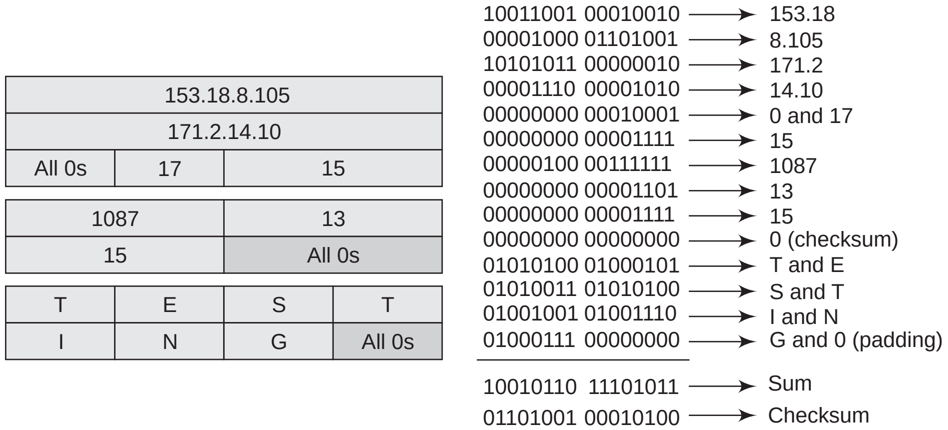


Figure 5.6(b): Checksum calculation in UDP

To calculate the checksum, take the following example. The pseudo header, the UDP header and data have values entered. The data is just 7 bytes in length. Since it is odd in number, padding of 0s is added in order to calculate the checksum.

The data is in the string or ASCII format. Its corresponding binary format is taken while computing the checksum. On adding the pseudo-header, header and data, the sum's one's complement gives the checksum (a 16-bit number).

The calculation and inclusion of the checksum into the datagram is optional. If the checksum is not computed, its field is filled in with 1s. When calculating the checksum, remember that the value cannot be all 1s as this would mean that the sum was all 0s (ones complement). That would be impossible as it would require all the fields in the datagram to be zeroes.

If the resultant checksum is all 0s, then the datagram is accepted. If the result is otherwise, it means that there has been an error during transmission.

5.3.5 UDP Procedures

The techniques and concepts used by UDP are common to the transport layer protocols. Some of the concepts are:

5.3.5.1 Connectionless Service

Connectionless service refers to the transmission of datagrams where each datagram is considered separate and independent. There is absolutely no relationship between the datagrams being sent even if they are from the same source or going towards the same destination. This means that the datagrams may travel via various routes. These datagrams lack numbering and the UDP service has done away with both connection establishment and connection termination.

The consequence of being a connectionless service is that the process employing UDP cannot send a data stream to UDP and anticipate UDP chopping up the data stream into datagrams. Instead, ensure that every request is of a small size so that it can fit into a user datagram. This implies that only processes that have to send small sized messages can use UDP.

5.3.5.2 Flow and Error Control

Flow control and windowing mechanisms are not present in the UDP protocol. This may cause an overflow of messages at the receiver end. Error control mechanisms in UDP is absent with only checksum providing some form of error control.

Therefore, the sender of the message will have no idea whether the message has been duplicated or lost during transmission. If the receiver realizes there has been an error via the checksum, it just throws the datagram away.

Since flow control and error control is lacking for UDP, it is up to the process using UDP to do some kind of flow and error control.

5.3.5.3 Encapsulation and Decapsulation

When messages are being sent from one process to another, it gets encapsulated by the UDP protocol at the sender's side and de-capsulated again by the UDP protocol at the destination. The following figure shows this process:

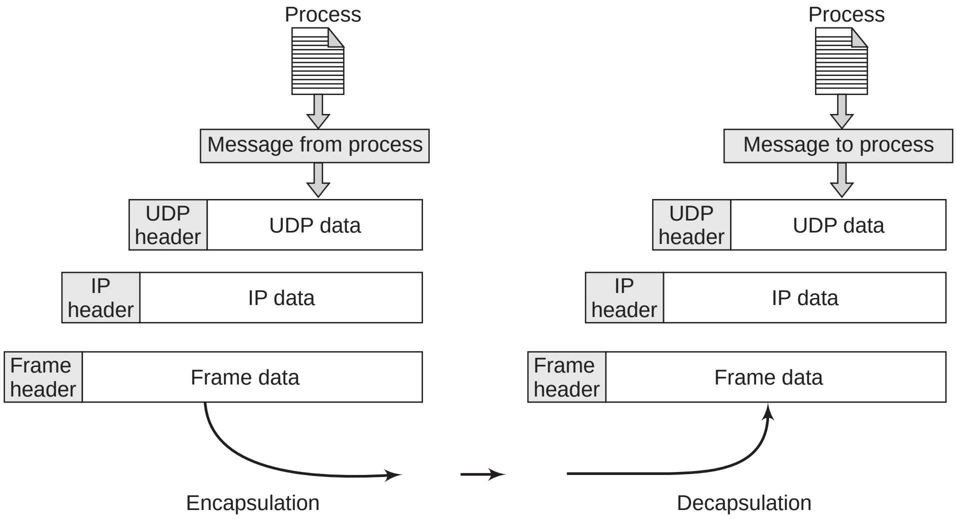


Figure 5.7: Encapsulation and decapsulation in UDP

In the *encapsulation* procedure, when the process wants to send the message via UDP, it is sent along with a pair of socket addresses and data length. On receiving the data, UDP adds its header and sends the user datagram onto the IP. IP will add its own header, and pass it along to the data link layer. IP also puts in a value=17 into the protocol field implying that the datagram has come from the UDP protocol and no other transport protocol.

The IP datagram continues its journey through the data link layer where it gets a data link header. Finally, it reaches the physical layer where the datagram's bits are encoded into optical or electrical signals (depending on the physical medium) and is transmitted to the destination.

In the *de-capsulation* procedure, the opposite occurs. The physical layer at the destination host gets hold of the message and decodes the signal into bits, which is passed onto the data link layer. This layer checks the header and trailer, and tries to detect any error.

If there is no error, the header and trailer are removed and IP gets the datagram next. IP does its own error checking, and if it could not find any, removes the header and passes the datagram to UDP.

Now, UDP uses checksum to confirm that there are no errors in the entire datagram. Again, if there are no errors, the header is removed and you are left with the data. This application data is sent along with the sender's socket address (needed if the process needs to respond back) to the process.

5.3.6 Uses of UDP

- Its lack of flow and error control makes UDP suitable for processes that merely need a simple request-response communication and are not concerned with flow and error control. Processes like FTP that send data in bulk cannot use UDP.
- Processes that have an existing internal flow and error control mechanism can use UDP. E.g. Trivial File Transfer Protocol (TFTP)
- UDP is suitable for multicasting as this property is existent in the UDP software but not in the TCP software.
- UDP can be used for management processes like SNMP.
- Route updating protocols like RIP (Routing Information Protocol) uses UDP as well.

5.4 TRANSMISSION CONTROL PROTOCOL (TCP)

Transmission Control Protocol (TCP) is a transport layer (Layer 4 of the OSI model) protocol. It ensures the *reliable* transmission of data. TCP is a connection-oriented protocol, which means that a virtual connection is made between two TCP's when exchanging data. Just like a telephone call, two communicating TCPs must first agree on the willingness to communicate. It uses flow and error control mechanisms, guarantees reliable and in-order delivery of data, full duplex operation, and multiplexing.

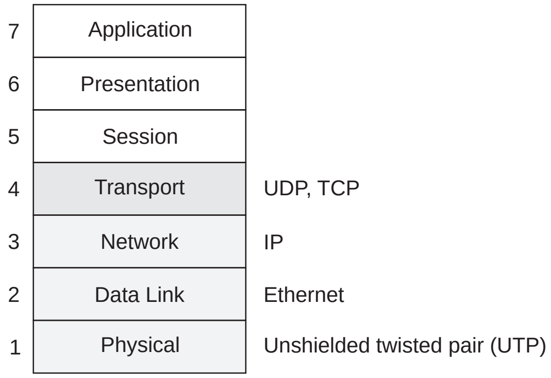


Figure 5.8: TCP architecture

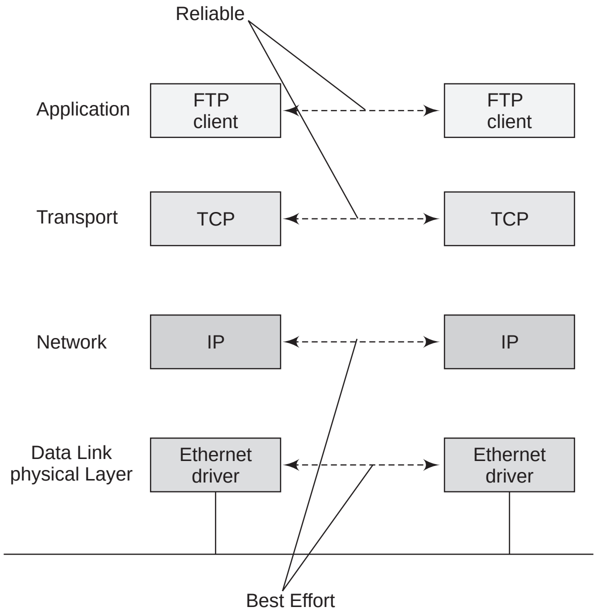


Figure 5.9: Reliability in TCP

The in-order delivery of data is achieved with TCP delivering an unstructured stream of bytes sequenced with a forwarding acknowledgement number. The bytes that are not acknowledged within a time frame are retransmitted. This ensures the reliability of data transfer as packets that are list, delayed, or misread are dealt with suitably.

5.4.1 Reliability Measures

TCP employs some measures by which it ensures reliability of data transfer. They are:

- **Checksums:** Every TCP segment has a checksum, which can be used by the receiver to determine if there has been some error in the header or the data.
- **Duplicate data detection:** In a packet switched network, there is a high possibility of packets getting duplicated which is why TCP keeps a track of bytes that have been received. In this way, it can reject packets that are just duplicate copies.
- **Retransmissions:** TCP implements retransmission of data if there is loss or damage to the data. The receiver needs to send acknowledgements back to the sender for each data packet received. So, if there is no acknowledgement and if the time out period is over, the data is retransmitted.
- **Sequencing:** Even if packets in the course of a packet switched network arrive out of order, the TCP is responsible for sequencing them before passing it onto the application.
- **Timers:** There are static and dynamic timers maintained for every data segment sent. If an acknowledgment does not come within the time stipulated by the timer, the sender will retransmit the data segment.

In addition, some other measures taken by TCP include:

- **Streams:** The TCP data is arranged as a stream of bytes. The fact that data is sent as datagrams in a network is concealed. The urgent pointer flags the out-of-band data.

  Since the TCP is in between the application layer and the network layer, when an application sends data to TCP, it sends it in the form of byte (8-bit) streams.

The TCP then decides whether it should segment the bytes or delineate the byte stream so that the data is sent in some manageable form. The total lack of "boundaries" gives TCP the apt name of being a *byte stream delivery service.*

- **Reliable delivery:** The data sent using TCP as the transport protocol is sequenced so that it is easy to determine the packets that have been transmitted or received. If data has been lost, TCP retransmits it.
- **Congestion Control:** TCP has the capability to dynamically determine the delay/loss characteristics of a network. It can thus adjust its operation (the number of packets sent, etc) in such a way that the throughput is maximized without the network getting overloaded.
- **Flow Control:** Unlike UDP, flow control mechanisms exist in TCP. Traffic is coordinated and data buffers are managed by TCP in such a way that the buffers never overflow. By the process of flow control, if there are any fast senders, they will be stopped at regular periods of time so that slow receivers can catch up.

TCP supports full duplex operation as TCP processes are able to send and receive simultaneously. The efficient flow control mechanism demands an acknowledgement be sent back to the sending host by the receiving TCP process with the highest sequence number that can be received with no danger of its internal buffers overflowing. Multiplexing of several upper-layer communications over a single connection is possible with TCP as well.

Just like the User Datagram Protocol (UDP), TCP uses port numbers in its communication and is a process-to-process protocol. Port numbers identify the exact process. The socket address is the combination of the IP address and the port number. Port numbers can be shared by both transport protocols—UDP and TCP. This means that a single port number can be assigned to two processes in the same host computer; but one process uses UDP and the other one uses TCP. To identify the protocol being used, read out the protocol field in the IP header.

5.4.2 TCP Services

TCP offers some services to the application layer processes. Let us discuss some of them in detail:

1. **Process-to Process Communication:** Just like UDP, TCP also provided a process-to-process communication with the help of port numbers.

   Some of the well-known ports that are used by TCP are listed in the following table:

Table 5.2: Well-known TCP ports

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP. Data	File Transfer Protocol (data connection)
21	FTP. Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol

53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

2. **Stream Delivery Service:** TCP is a stream-oriented protocol, quite unlike UDP. The data stream received by the destination process is in the same manner as it was sent by the sending process. The stream delivery service offered by TCP gives an impression of an imaginary *tube* connecting the two processes. The following figure depicts the imaginary environment that TCP creates between the sending and receiving processes.

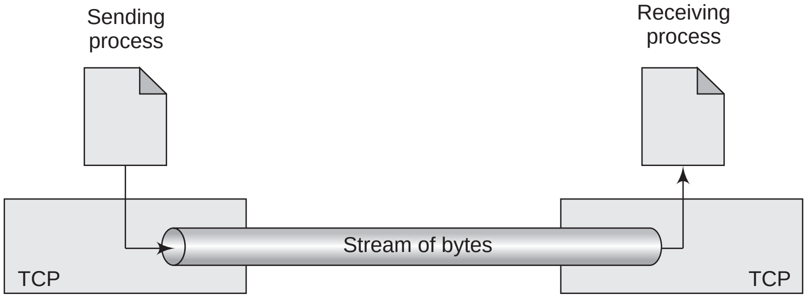


Figure 5.10(a): Stream delivery in TCP

Sending and Receiving Buffers

For stream delivery, TCP uses buffers for storage as the sending and receiving processes may not write onto the stream or read from the stream at the same speed. There will be two buffers: the sending buffer and the receiving buffer. The buffer shown in the figure is a circular array of 1 byte locations. In reality, the buffers will be large of at least hundreds or thousands of bytes in size.

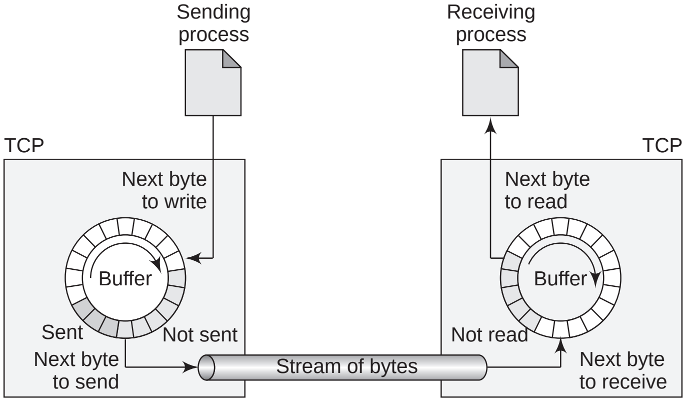


Figure 5.10(b): Stream delivery using buffers

The sending buffer stores the data that is coming from the sending application program. This data is sent from the program to the buffer at the same rate as it is created. The program, using a write operation, writes the data to the buffer and may unite the output of multiple write operations into one segment before transmission.

The receiving TCP gets segments after transmission and stores them in the receiving buffer. The read operation is used to read the data from the buffer. Until the receiving application reads the data completely, it is kept in the buffer. This is mostly helpful if the reading rate is slower than the receiving rate.

5.4.3 TCP Features

The main features of TCP would be its numbering system, flow and error control techniques, and the congestion control feature.

1. **Numbering System:** TCP tracks how many bytes of data is transmitted or received with the help of two numbers: ***sequence number*** and ***acknowledgement number***. Note that these two numbers are byte numbers and not segment numbers.

   Byte Number: All bytes involved in a transmission are numbered by TCP. The numbering is done when bytes of data are received and stored in the sending buffer. The numbering is independent in both directions. TCP numbers the first byte with a random number between 0 and $2^{32} - 1$. This byte numbering technique is used in flow and error control too.

   Sequence number: Once the bytes have been numbered, a sequence number is allocated to every segment being sent. This sequence number for a segment would be the first byte's number in that segment.

   e.g.: Segment 1 with sequence number 3001 (range: 3001 to 4000).

   Acknowledgement number: Two hosts communicating with each other, will number their bytes with a different starting byte number. The sequence number will indicate the number of the first byte being carried by that segment. Now, an acknowledgement number is employed to indicate the bytes it has received. This number will also indicate the next byte the receiving party expects to receive.

   This acknowledgement number is *cumulative,* where the receiving party extracts the number of the last byte it has received, adds 1 to it, and says that this is its acknowledgement number. *e.g*. if the last byte received safe and sound is 5006, then the acknowledgement number would be 5007, indicating that it has received all bytes from the beginning (need not be from 0) till 5006.
2. **Flow Control:** In flow control, the receiving station has a control over how much data is being sent to him so that there is no overflow of data thus overwhelming the receiver. Employing the byte oriented numbering system ensures the flow control to be byte-oriented as well.
3. **Error Control:** Though error control is byte oriented as well, TCP utilizes a segment as a unit of data for error detection.
4. **Congestion Control:** The data being sent by the receiver is controlled by two factors: the receiver, so that there is no overflow and by the congestion level in the network.

5.4.4 TCP Segment

The unit of data in IP is referred to as an IP datagram. For UDP, it is known as a UDP datagram, but when it comes to TCP, the unit of data is known as the TCP segment. The header can range from 20 – 60 bytes. It would be 20 bytes if there are no options, and it would e 60 bytes if the options are present.

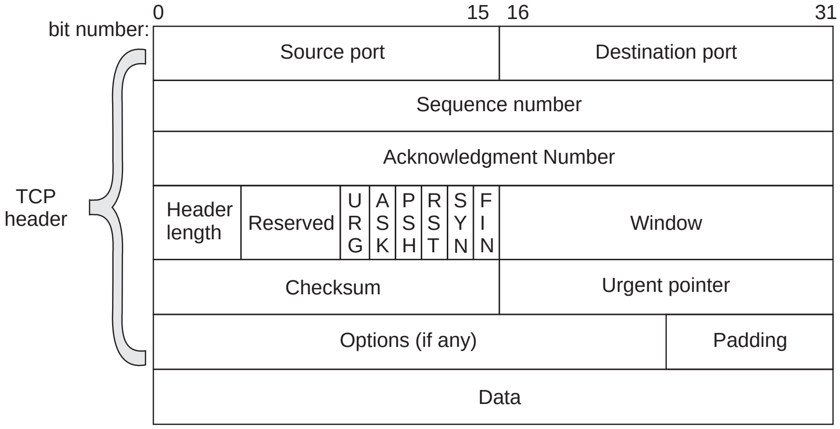


Figure 5.11(a): TCP segment

5.4.4.1 TCP Segment Field Descriptions

The TCP packet will have a 20-60 byte header followed by the data. If there are no options, the header will be only 20 bytes long. Otherwise, it is 60 bytes long. The various fields are:

- **Source port address (16 bits):** Identifies the port number of the application program (upper layer source process) in the source computer that is sending the segment.
- **Destination port address (16 bits):** Defines the port number of the application program in the destination computer that will be receiving the segment.
- **Sequence number (32 bits):** This field specifies the number given to the first byte of data in the current segment. This helps in the numbering of each byte that is going to be transmitted. This sequence number lets the receiver know which byte is the first one in the segment. During the connection establishment phase, this field identifies the initial sequence number.
- **Acknowledgement number (32 bits):** Will contain the sequence number of the next byte that the receiver of the segment is expecting from the other communicating party. If the receiver of the segment has received byte number y, it will set the acknowledgement number to be y+1. It is possible to send the acknowledgement and data together.
- **Header length (HLEN- 4 bits):** Contains the number of 32 bit (4 byte) words in the TCP header. Since the TCP header can be anywhere in between 20-60 bytes in length, HLEN's value could be between 5 (5 x 4=20) and 15 (15 x 4 = 60).
- **Reserved (6 bits):** Reserved for future use.
- **Control (6 bits):** This field contains 6 flags or control bits which can set. The various flags like SYN and ACK are used for connection establishment, while FIN is used for termination etc. The following figure shows all the flags:

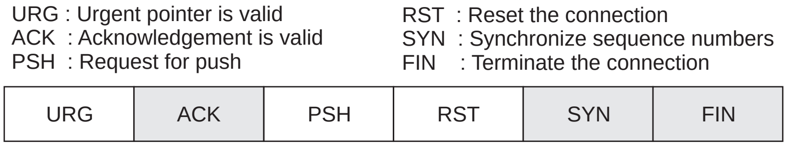

URG : Urgent pointer is valid
ACK : Acknowledgement is valid
PSH : Request for push
RST : Reset the connection
SYN : Synchronize sequence numbers
FIN : Terminate the connection
URG
ACK
PSH
RST
SYN
FIN

Figure 5.11(b): Control flags

- **Window Size (16 bits):** Indicates the size, in bytes, of the window the other party must maintain for incoming data. Since this field has a length of 16 bits, the maximum window size can be for 65,535 bytes. It is the receiver of a segment that decides the window size.

- **Checksum (16 bits):** This field will indicate if the header was damaged during the transmission. It is mandatory that this field be included in the TCP segment whereas it was optional in a UDP datagram.
- **Urgent pointer (16 bits):** This field is valid only if the urgent flag is set. It contains the number to be added to the sequence number so that the last urgent byte in the segment can be obtained.
- **Options (up to 40 bytes):** Various TCP options are specified here.

5.4.5 TCP Connection

Since TCP is a connection-oriented transport protocol, a virtual path is established between the source and the destination. Once this single virtual path is made, all the segments that are part of one message can be sent across it. Retransmission, acknowledgements etc happen over this pathway as well.

The TCP connection is done by two procedures: connection establishment and connection termination.

1. TCP Connection Establishment

TCP (a connection oriented protocol) employs IP (which is a connection-less protocol) in the data communication. Though this sounds strange, it is achieved because the TCP connection made is virtual and not physical. TCP just uses IP in the sending process but the reins of control is with TCP itself. Say, a segment is lost or damaged. It is then retransmitted. This information is known to TCP only; IP is unaware of this retransmission. Similarly, if segments arrive out of order, TCP will hold on to the segments until the remaining ones arrive and IP is left unaware of the reordering that is happening.

To have a reliable transport service, the TCP hosts who want to communicate must first establish a connection-oriented session with each other. TCP transmits data in the full-duplex mode, which means that once connected, the two TCP hosts can transmit segments simultaneously to each other.

Connection establishment is performed by using the ***three-way handshake*** mechanism. This includes each communicating party initializing the communication and obtaining approval from the other side before any transfer of data. Both sides also consent to initial sequence numbers. The three way handshake is necessary to ensure that no data is transmitted or even retransmitted during the connection establishment phase or after the session has been terminated.

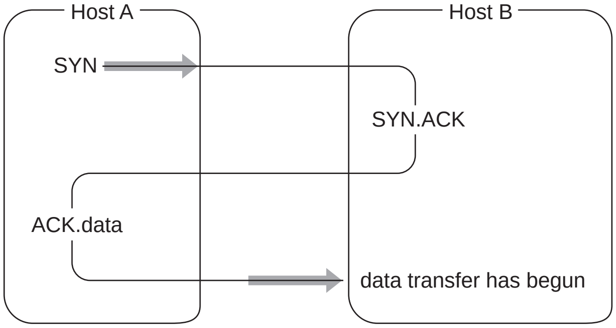


Figure 5.12(a): TCP connection

The following diagram shows the three way handshake where three segments are exchanged between Hosts A and B.

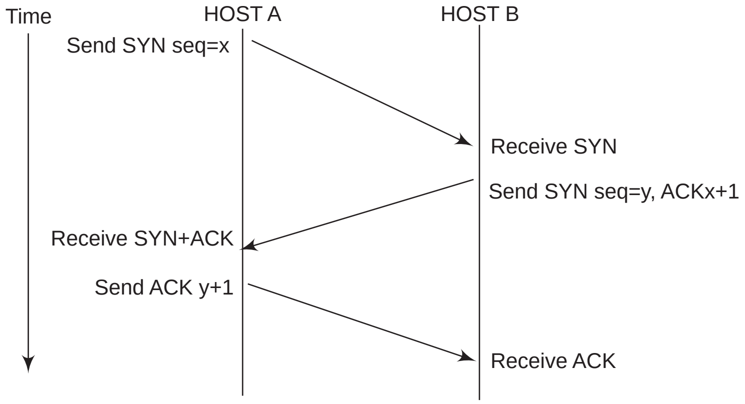


Figure 5.12(b): Three-way handshake for connection establishment

Each host will choose a sequence number randomly which will help in monitoring the bytes being sent and received. The three-way handshake procedure is as follows:

- In the beginning, Host A initiates the connection by sending a ***SYN segment*** with initial sequence number = X and SYN bit set thus showing that this TCP segment is a connection request. It synchronizes the sequence numbers.
- Host B will receive the SYN segment, process it and replies with an acknowledgement to the connection request. Two bits are set: SYN and ACK making it a ***SYN+ACK segment.*** The initial sequence number = Y, ACK bit is set to X+1. This means that Host B expects the next byte from A to be data with a sequence number of X+1. This method is known as ***forward acknowledgement*** where if ACK=1000, it means B has already received bytes from 0-999 and expects byte 1000 next.
- Host A will now have received B's SYN+ACK. The final connection establishment step is completed with A sending a final acknowledgement segment with ACK set, and the acknowledgement field= Y+1. This is just an ***ACK segment***. This lets host B know that the next byte A expects is with sequence number Y+1. The sequence number would be X+1. Data transfer can now begin.

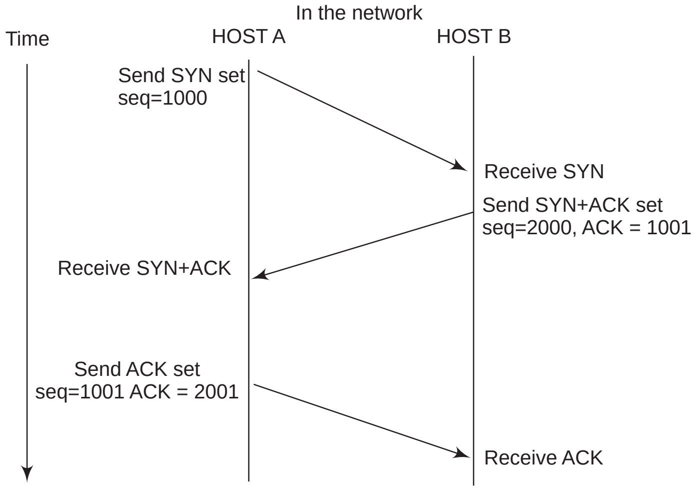


Figure 5.12(c): Three-way handshake using sequence number

Positive Acknowledgement and Retransmission (PAR)

In very simple terms, a host using positive acknowledgement and retransmission (PAR) will send the data again till it gets the message from the destination that the data has arrived.

The reliability and flow control technique implemented by transport protocols works with a timer. Once a packet is sent, the timer is started, and before the time runs out, an acknowledgement from the remote station must be received before another packet can be sent out. If the timer expires before the acknowledgement arrives, the packet is retransmitted.

PAR can track if packets have been lost or duplicated (which can happen aplenty over a packet switched network due to delays) by the sequence number assigned to every packet. This sequence number will be included in the acknowledgement sent back to the sender thus allowing the TCP process to track the acknowledgements as well.

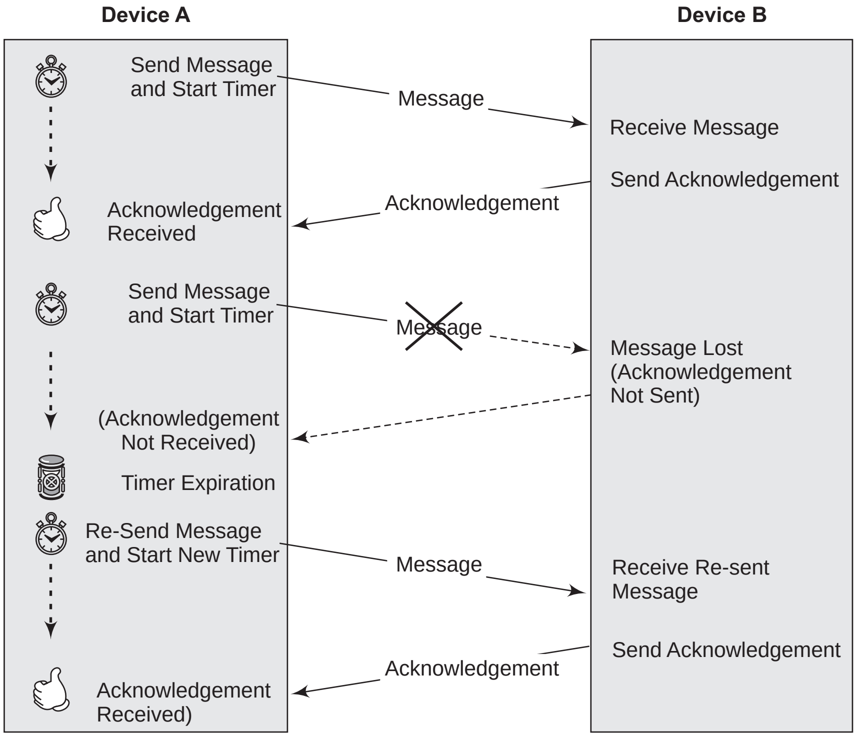

Device A
Device B
Send Message and Start Timer
Message
Receive Message
Send Acknowledgement
Acknowledgement
Acknowledgement Received
Send Message and Start Timer
Message
Message Lost (Acknowledgement Not Sent)
(Acknowledgement Not Received)
Timer Expiration
Re-Send Message and Start New Timer
Message
Receive Re-sent Message
Send Acknowledgement
Acknowledgement
Acknowledgement Received)

Figure 5.13: Reliable transmission using PAR

With all the advantages of amazing data transmission reliability, PAR does sound like an inefficient use of bandwidth as the host will have to wait for an acknowledgement packet each time and then only send the next packet, thus allowing a single packet transmission at one time.

2. Connection Termination

Any of the two parties involved in the data communication can terminate the connection between them. A total of four steps will be needed to terminate the connection:

- X (mostly, the client) will send a segment saying it wants to terminate the connection. This is a request for an ***active close.***
- Y will acknowledge X’s request for termination with a segment of its own. Now, the connection is closed in the X-Y direction but Y can still send data if it wants to.

- Once Y has finished sending data, it can send a segment saying it wants the connection terminated too. This is the ***passive close.***
- Now, X will confirm the request from Y.

The above connection termination is known as ***four-way handshaking***.

5.4.6 Flow Control Mechanism in TCP

As already discussed, flow control in TCP specifies how many bytes of data the source can send before it receives an acknowledgement from the receiver. The slowest process would be sending one byte of data and waiting for an acknowledgement before sending the next byte. This would seem impractical especially if the data has to cover a large distance before reaching its destination, which would mean that the sender is idle till the time that the acknowledgement comes back.

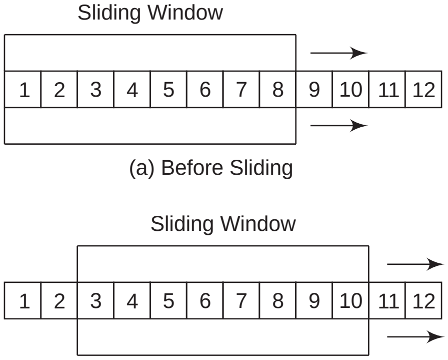
The faster method would be to send all data and then wait for an acknowledgement. Though the data transfer is much faster now, the receiver may be overwhelmed by the amount of data coming in. Also, if any part of the data has been lost or duplicated, the sender would not know unless the receiver checks the entire message.

That's how the concept of *sliding window* came up which was an in-between solution.

5.4.6.1 Sliding Window

Flow control in TCP is achieved by the sliding window protocol. A window covers that part of the data that can be sent by the host before having to receive an acknowledgement. This window will slide over the data in the buffer as and when acknowledgements come in indicating that transmission has occurred with no error.

A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data. TCP sliding windows are byte-oriented



(a) Before Sliding

(b) After Sliding

Figure 5.14(a): Sliding window

The above figure shows the case where the sliding window is of size = 8. This means that the sender can transmit 8 bytes before waiting for an acknowledgment. Now, if acknowledgement for the first 2 bytes comes in, then the window can slide two bytes to the right. Now, it can again transmit 8 more bytes before needing to wait for any acknowledgement. The sender will have a pointer that lets it know how many bytes have been sent and how many are left to be sent.

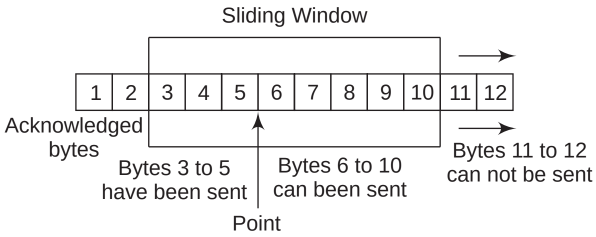
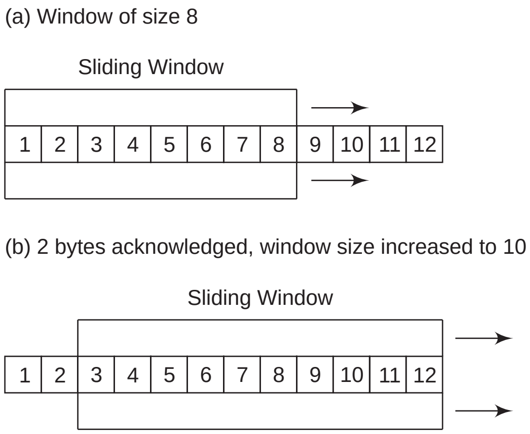

Sliding Window
1
2
3
4
5
6
7
8
9
10
11
12
Acknowledged bytes
Bytes 3 to 5 have been sent
Bytes 6 to 10 can been sent
Bytes 11 to 12 can not be sent
Point

Figure 5.14(b): Description of bits in sliding window

Increasing the Window Size

The window size is not a fixed one. It can be varied so that flow control can be attained. It is the receiving station that sets the window size when it sends the acknowledgement segment. The next figure shows how the window size has been increased from 8 to 10.


(a) Window of size 8

(b) 2 bytes acknowledged, window size increased to 10

Figure 5.14(c): Increasing the window size

In the same way, the window size can be decreased as well again on the behest of the receiver. The only restriction here is that when decreasing the window size, the leasing edge of the window cannot slide to the left.

Window Management

As discussed before, the receiver determines the sliding window size in the sending TCP and informs the sender via the ACK segments. The receiver decides on the sliding window size based on the space remaining in its receiving TCP buffer.

We presume that the receiving TCP has defined a buffer size of 1000 bytes and it announces the window size to be 1000. The sending TCP therefore sends 1000 bytes in its first segment. The buffer becomes full. The receiver will acknowledge the segment but also define a window size of zero as it cannot receive any more segments.

The sending TCP waits until it gets an acknowledgement with a window size greater than zero. Now, when the application program consumes 500 bytes, and there is 500 bytes worth of space in the buffer, it sets the window size to be 500 bytes in a new acknowledgement to the sending TCP. The sender now transmits a segment with 500 bytes of data.

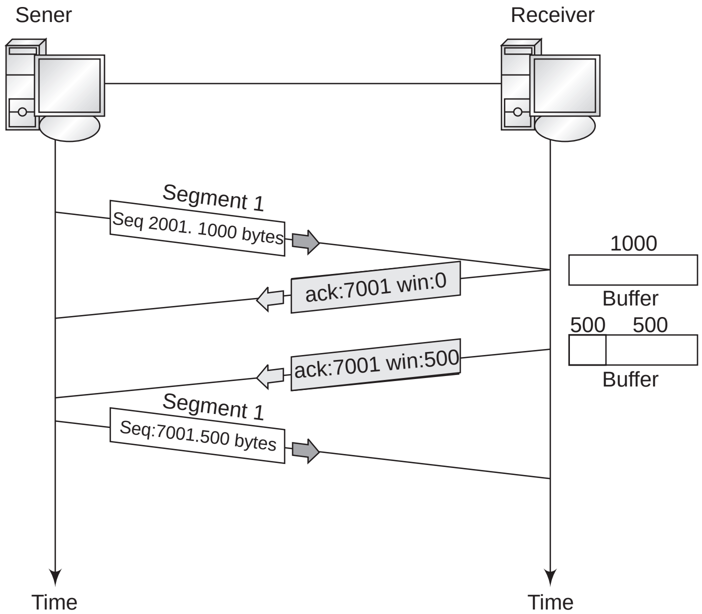


Figure 5.15: Flow control by window size management

In this way, by adjusting the window size, flow control is managed and controlled.

5.4.7 Error Control Mechanism in TCP

TCP is a reliable transport protocol as well. Application programs can rely on TCP for the delivery of data with no error, no lost packets and in-order. This reliability is provided by the error control mechanisms seen in TCP for determining corrupted segments, lost segments, out-of-order segments, and duplicated segments. Once detected, the errors are corrected as well.

The three tools used in error detection are:

1. **Checksum:** The checksum field in every segment verifies the segment. It is computed at both the sender and destination ends and if the destination realizes that the segment is corrupted, it throws the segment away.
2. **Acknowledgement:** TCP uses the acknowledgement technique to confirm whether segments have been received by the destination TCP. If not, it assumes that the segments have been lost, or corrupted.
3. **Time-out:** Refers to the time the sending TCP waits until it gets an acknowledgement, after which, it assumes that the sent packet is either lost, or corrupted and retransmits it.

As discussed previously, the sending TCP will have a timer for every segment that is being sent. If no acknowledgment arrives before the timer count-down expires, then the segment is considered to be either lost or corrupted and is retransmitted.

Lost Segment

Lost segments are similar to corrupted segments from the point of view of the sender and the destination. While a corrupted segment is discarded by the final destination, a lost segment will have been discarded by some node during the course of transmission and will never reach the final destination.

Duplicate Segment

Duplicate segments are mostly sent out by the sending TCP if it has not received the acknowledgement for the segment before the timer expires. However, since the segments are numbered, and since TCP expects the bytes to in sequence when they arrive, if a packet arrives with the same sequence number as one that has already been received, then the destination TCP simply discards the duplicate segment.

Out-of-Order Segment

TCP, at the transport level utilizes the services of IP, the protocol at the network layer, which is an unreliable and connectionless protocol. As the TCP segment is passed onto the network layer, it is encapsulated in an IP datagram. Routers send the datagrams via any route. This causes datagrams to reach the destination out-of-order as some routes may be shorter and some longer.

The receiving TCP, on seeing out of order segments, does not acknowledge them unless the segments that precede them arrive as well. If sending the acknowledgement does not happen before the timer expires, then the segment will be retransmitted. In such cases, the destination TCP will discard the duplicate segments.

Lost Acknowledgement

In the process of TCP acknowledgements, it is possible that the source TCP does not realize that an acknowledgement has been lost. The reason why this can happen is because TCP utilizes a *cumulative acknowledgement* system wherein an acknowledgement confirms that all bytes up to the byte indicated in the acknowledgement has been received. Say, if the destination TCP has sent an acknowledgement with an ACK=3001, then it would mean that bytes from 1001-3000 have been received (assuming that the first byte sent had the initial byte number of 1001). If the next ACK=5001, then it means that the destination has received all bytes from 1001-5000. So, even if the previous acknowledgment=3001 was lost, it does not cause a difference at all.

SUMMARY

- The Transport Layer in the TCP/IP protocol model provides the mechanism for process-to-process communication.
- The transport layer performs mainly four functions: Segmenting and assembling upper-layer applications, Transport of segments from host-to-host, Establishment and management of end-to-end operations, and Error recovery.
- The User Datagram Protocol (UDP) is the ***connection-less, unreliable*** transport layer (Layer 4) protocol.
- UDP is a *lightweight* layer above the Internet Protocol with a small overhead and a very simple protocol. This protocol can be used in those situations where a small message has to be sent and reliability is not a deciding factor.
- To uniquely identify the network application process running on the host machine that requires TCP/IP communication, a 16-bit *port number* is assigned to it.
- The client process/program is given a random port number by the UDP software running on the client host. This port number is referred to as the *ephemeral port number.*

- The IP address and the port number together comprise the socket address.
- The UDP checksum includes three parts: Pseudo header, UDP header, Data
- Transmission Control Protocol (TCP) is a transport layer (Layer 4 of the OSI model) protocol. It ensures the *reliable* transmission of data and is a connection-oriented protocol.
- TCP tracks how many bytes of data is transmitted or received with the help of two numbers: ***sequence number*** and ***acknowledgement number***.
- The TCP connection is done by two procedures: connection establishment and connection termination.
- Connection establishment is performed by using the ***three-way handshake*** mechanism.
- For a more efficient use of network bandwidth, a ***TCP sliding window*** was introduced which allows for multiple bytes or packets to be sent before waiting for an acknowledgment.
- The three tools used in error detection are: Checksum, Acknowledgement and Time-out.

6

WORLD WIDE WEB

Introduction

The application layer is responsible for providing services to the user. The application layer enables the user or software to access the network. It provides user interfaces. It supports services such as electronic mail, file access and transfer, access to system resources, surfing the World Wide Web, and network management.

A distributed application is a program that runs on more than one computer and communicates through a network. There are two different software programs available in distributed applications; the back-end (server) software and the front-end (client) software. Back-end software runs on a shared system and manages shared resources such as disks, printers, and modems. The front-end software runs on workstations. It handles user interface, such as getting input from a keyboard and displaying output to a screen.

Distributed applications can be simple with a single client computer and a single server or more complex, allowing many client computers and several servers. For example, web browsers are distributed applications. Browsers need back-end software that is server on the World Wide Web and front-end software installed on the workstation such as Netscape Communicator or Internet Explorer.

6.1 WORLD WIDE WEB (WWW) AND HTTP

The World Wide Web (WWW), or simply Web, is a global network of servers linked by a common protocol allowing access to all connected hypertext resources. When a client host requests an object, a Web server responds by sending the requested object through browsing tools.

A browser is a user agent displaying the requested Web page. The Hyper Text Transfer Protocol (HTTP) transfers that page at the application layer. HTTP uses TCP rather than UDP, since reliability of delivery is important for Web pages with text. The TCP connection-establishment delay in HTTP is one of the main contributing delay factors associated with downloading Web documents.

HTTP is based on the client/server idea, having a client and a server program, both of which can be executed on different end systems. The communication is carried out through an exchange of HTTP messages. This protocol specifies the structure of these messages. For example, HTTP defines how a pair of client/server hosts should exchange messages. In this context, a Web page consists of files, such as Hypertext Markup Language (HTML) file or an image that can be addressed by a single uniform resource locator (URL). A URL is a global address of an HTML document and has two parts. The first part indicates what protocol is used, and the second part determines the IP address of the associated resource.

6.1.1 Web Caching (Proxy Server)

An HTTP request from a user is first directed to the network proxy server, or Web cache. Once configured by the network, a browser's request for an object is directed to the Web cache, which must contain updated copies of all objects in its defined proximity. The main reason for Web caching is to reduce the response time for a user request. This benefit is much more obvious when the bandwidth to a requested server is limited because of traffic at certain hours of the day. Normally, each organization or ISP should have its own cache providing a high-speed link to its users. Consequently, it is to users' advantages that this rapid method of finding objects be available. This method of Internet access also reduces traffic on an organization's access link to the Internet. The details of Web caching are as follows:

Begin Web Caching Algorithm

1. The source browser makes a TCP connection to the Web cache.
2. The user browser transmits its HTTP request to the Web cache.
3. If it has a copy of the requested object, the Web cache forwards the object to the user browser.

Otherwise the Web cache establishes a TCP connection to the requested server and asks for the object. Once it receives the requested object, the Web cache stores a copy of it and forwards another copy to the requesting user browser over the existing TCP connection.

Figure shows three Internet service providers (ISPs). A user in ISP domain 3 is browsing to find and watch an object named http://www.filmmaker.com in ISP domain 1. The request for this object is directed to the Web cache, shown by dashed lines. In this example, the Web cache has no record of the requested object and therefore is establishing another TCP connection to update its record.

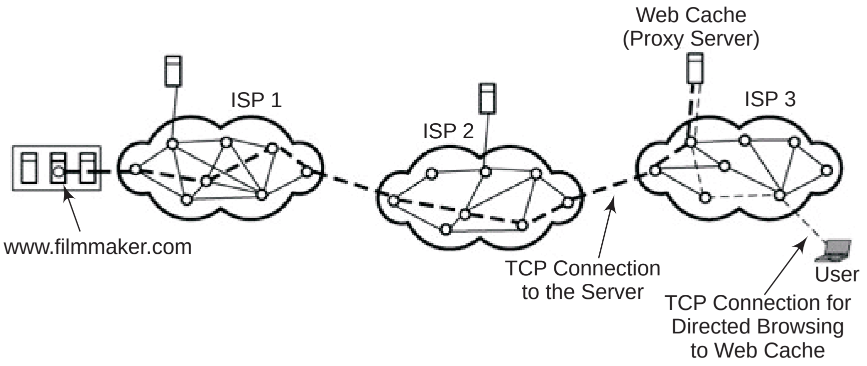


Figure 6.1: A user's browser requesting an object through the Web cache

6.1.2 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is used to access data on the World Wide Web. HTTP is not a protocol for transferring hypertext but it is a protocol for transmitting information with the efficiency needed for making hypertext jumps. The data transferred by the protocol can be plain text, hypertext, audio, images, or any Internet-accessible information.

It functions like FTP as well as like SMTP. It transfers files and uses the services of TCP like FTP. While compare to FTP, it uses only one TCP connection hence it is more simple. The data transferred between the client and the server look like SMTP messages and the format of the messages is controlled by headers like MIME.

The difference between SMTP and HTTP is, the messages are read and interpreted by the HTTP server and HTTP client (browser). SMTP messages are stored and forwarded, but HTTP messages are delivered immediately. HTTP uses the services of TCP.

6.1.2.1 Transaction

The figure below explains the HTTP transaction between the client and server. The client initializes the transaction by sending a request message. The server replies by sending a response. The formats of the request and response messages are similar.

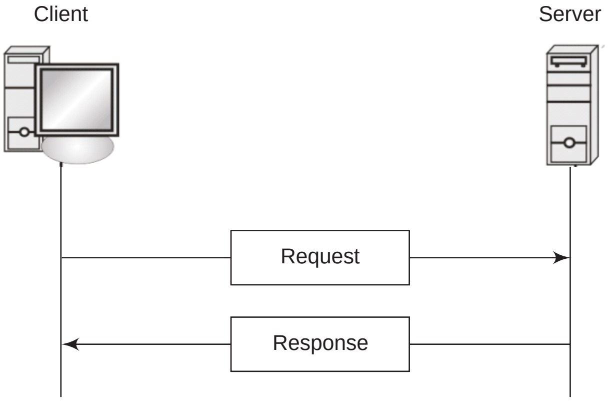


Figure 6.2: HTTP transaction between client and server

There is no end-to-end TCP connection between the client and the server. There are one or more intermediate systems with TCP connection between logically adjacent systems. Intermediate system acts as a relay hence a request initiated by the client is relayed through the intermediate systems to the server and the response from the server is relayed back to the client. Three forms of intermediate system are defined in the HTTP specification. They are,

- **Proxy** – A proxy operates for other clients and presents requests from other clients to a server. The proxy acts as a server in interacting with a client and as client in interacting with a server.
- **Gateway** – A gateway is a server that appears to the client as if it was an origin server. It acts on behalf of other servers that may not be able to communicate directly with a client.
- **Tunnel** - The tunnel does not work on HTTP requests and responses. It is simply a relay point between two TCP connections and the HTTP messages are passed unchanged—as if there was a single HTTP connection between client and server. Tunnels are used when there must

be an intermediary system between client and server but it's unnecessary for that system to understand the contents of any messages. For example; a firewall in which a client or server external to a protected network can establish an authenticated connection, and then maintain that connection for purposes of HTTP transactions.

6.1.2.2 Request Messages

We need to explain the individual elements of the HTTP message to describe the functionality of HTTP. HTTP consists of two types of messages: requests from clients to servers, and responses from servers to client. The general structure of the messages is shown in the figure below.

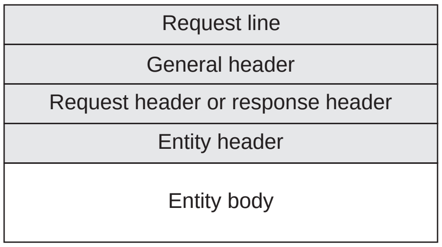


Figure 6.3: Structure of a request message

The following fields are used with full requests and responses,

- **Request-Line –** Identifies the message type and the requested resource.
- **Response-Line** – Provides status information about this response.
- **General-Header –** Contains fields that are applicable to both request and response messages but that do not apply to the entity being transferred.
- **Request-Header –** Contains information about the request and the client.
- **Response-Header** – Contains information about the response.
- **Entity-Header –** Contains information about the resource identified by the request and information about the entity body.
- **Entity-Body –** The body of the message.

6.1.2.3 Request Methods

HTTP defines eight methods.

- **HEAD –** This request is like a GET, except that the server's response must not include an entity body; all of the header fields in the response are the same as if the entity were present. This enables a client to get information about a resource without transferring the entity body.
- **GET –** A request to retrieve the information identified in the URL and return it in an entity body.
- **POST –** Submits data to be processed, for example from a HTML form to the identified resource. The data is included in the body of the request.
- **PUT –** A request to accept the attached entity and store it under the supplied URL.
- **DELETE –** Deletes the specified resource.
- **TRACE –** Sends back the received request, so that a client can see what intermediate servers are adding or changing in the request.
- **OPTIONS –** A request for information about the options available for the request/ response chain identified by this URL.
- **CONNECT –** Used with a proxy that can change to being an SSL tunnel.

6.2 DOMAIN NAME SYSTEM (DNS)

One of the most important components of the application layer is the Domain Name System (DNS) server. DNS is a distributed hierarchical and global directory that translates machine or domain names to numerical IP addresses. DNS can be thought as a distributed database system used to map host names to IP addresses, and vice versa. DNS is a critical infrastructure, and all hosts contact DNS servers when they initiate connections. DNS can run over either UDP or TCP. However, running over UDP is usually preferred, since a fast response for a transaction provided by UDP is required. Some of the information-processing functions a DNS server handles are:

- Finding the address of a particular host
- Delegating a subtree of server names to another server
- Denoting the start of the subtree that contains cache and configuration parameters, and giving corresponding addresses
- Naming a host that processes incoming mail for the designated target
- Finding the host type and the operating system information
- Finding an alias for the real name of a host
- Mapping IP addresses to host names

DNS is an application-layer protocol, and every Internet service provider whether for an organization, a university campus, or even a residence has a DNS server. In the normal mode of operation, a host sends UDP queries to a DNS server. The DNS server either replies or directs the queries to other servers. The DNS server also stores information other than host addresses.

The DNS routinely constructs a query message and passes it to the UDP transport layer without any handshaking with the UDP entity running on the destination end system. Then, a UDP header field is attached to the message, and the resulting segment is passed to the network layer. The network layer always encapsulates the UDP segment into a datagram. The datagram, or packet, is now sent to a DNS server. If the DNS server does not respond, the fault may be UDP's unreliability.

6.2.1 Domain Name Space

Any entity in the TCP/IP environment is identified by an IP address, which thereby identifies the connection of the corresponding host to the Internet. An IP address can also be assigned a domain name. Unique domain names assigned to hosts must be selected from a name space and are generally organized in a hierarchical fashion.

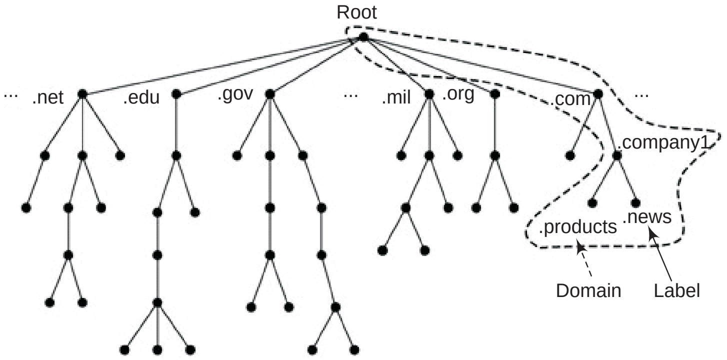


Figure 6.4(a): Hierarchy of domain name space, labels, and domain names

Domain names are defined in a tree-based structure with the root at the top, as shown in Figure 6.4(a). A tree is structured with a maximum of 128 levels, starting at level 0 (root). Each level consists of nodes. A node on a tree is identified by a label, with a string of up to 63 characters, except for the root label, which has empty string.

The last label of a domain name expresses the type of organization; other parts of the domain name indicate the hierarchy of the departments within the organization. Thus, an organization can add any suffix or prefix to its name to define its host or resources. A domain name is a sequence of labels separated by dots and is read from the node up to the root. For example, moving from right to left, we can parse as follows: domain name news.company1.com, a commercial organization (.com) and the "news" section of "company1" (news.company1). Domain names can also be partial. For example, company1.com is a partial domain name.

6.2.2 Domain-Name Servers

The domain name space is divided into subdomains, and each domain or subdomain is assigned a domain name server. This way, we can form a hierarchy of servers, as shown in Figure 6.4(b), just as we did for the hierarchy of domain names. A domain name server has a database consisting of all the information for every node under that domain. Each server at any location in the hierarchy can partition part of its domain and delegate some responsibility to another server. The root server supervises the entire domain name space. A root server typically does not store any information about domains and keeps references only to servers over which it has authority. Root servers are distributed around the world.

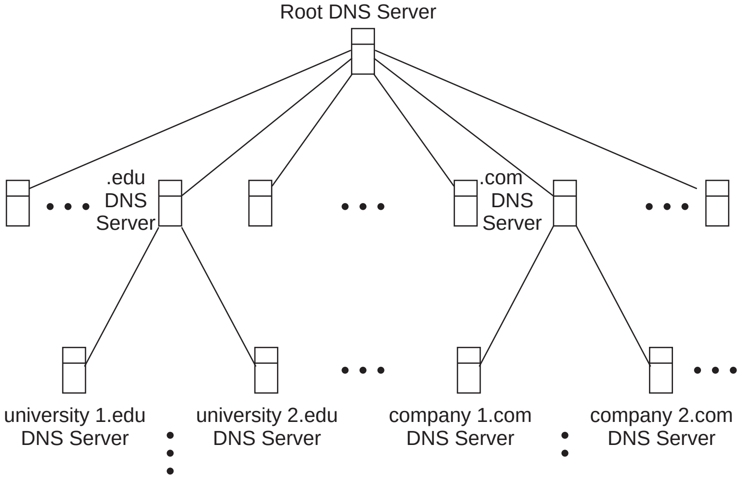


Figure 6.4(b): Hierarchy of DNS domain name servers

6.2.3 Name/Address Mapping

DNS operates based on the client/server application. Any client host can send an IP address to a domain name server to be mapped to a domain name. Each host that needs to map an address to a name or vice versa should access the closest DNS server with its request. The server finds and releases the requested information to the host. If the requested information is not found, the server either delegates the request to other servers or asks them to provide the information. After receiving the mapping information, the requesting host examines it for correctness and delivers it to the requesting process.

Mapping can be of either recursive or iterative. In recursive mapping Figure 6.4(c), the client host makes the request to its corresponding DNS server. The DNS server is responsible for finding the answer recursively. The requesting client host asks for the answer through its local DNS server, news.company1.com. Assume that this server contacts the root DNS server, and still the information has not been found. This time, the root DNS server sends the query to the .com server, but the transaction still remains unsuccessful. Finally, .com server sends the query to the local DNS server of the requested place, as dns.company2.com, and finds the answer. The answer to a query in this method is routed back to the origin, as shown in the figure. The local DNS server of the requested place is called the authoritative server and adds information to the mapping, called time to live (TTL).

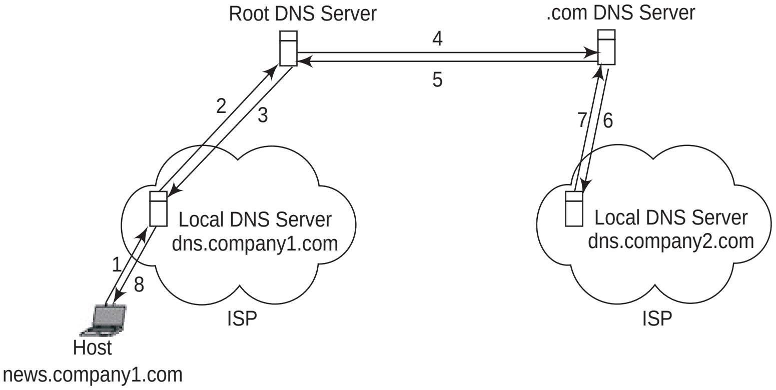


Figure 6.4 (c): Recursive mapping

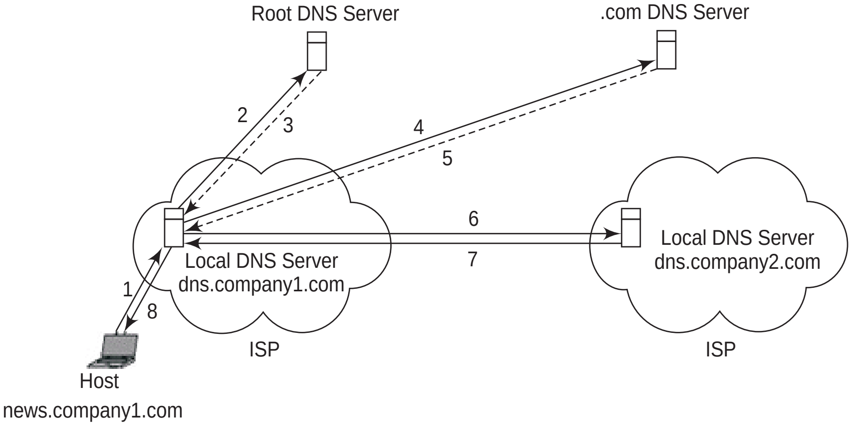


Figure 6.4 (d): Iterative mapping

The host must then repeat the query to the next DNS server that may be able to provide the name. This continues until the host succeeds in obtaining the name. In Figure 6.4(d), the news.company1.com host sends the query to its own local DNS server, dns.company1.com, thus trying the root DNS server first and then tries.com server, finally ending up with the local DNS server of the requested place: dns.company2.com.

6.2.4 DNS Message Format

DNS communication is made possible through query and reply messages. Both message types have the 12-byte header format shown in Figure 9.6. The query message consists of a header and a question message only. The reply message consists of a header and four message fields: question, answer, authority, and additional information.

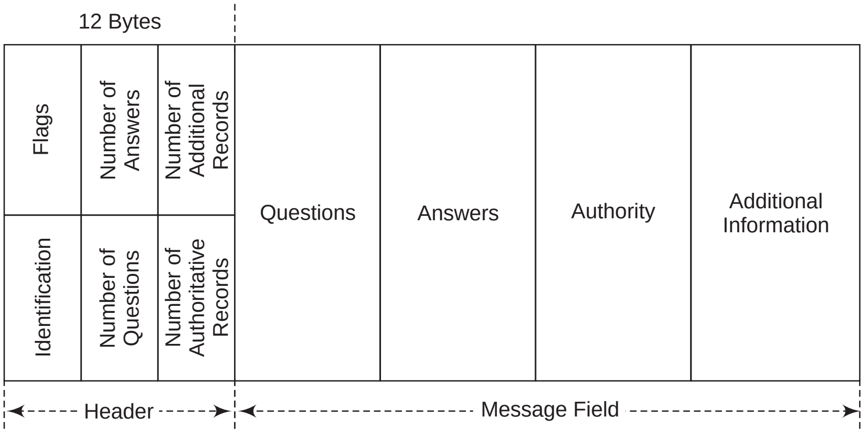


Figure 6.5: DNS message format

The header has six fields as follows. A client uses the identification field to match the reply with the query. This field may appear with a different number each time a client transmits a query. The server copies this number in its reply. The flags field contains subfields that represent the type of the message, such as the type of answer requested or requested DNS recursive or iterative mapping. The number of questions field indicates how many queries are in the question portion of the message. The number of answers shows how many answers are in the answer field. For the query message, this field contains all zeros. The number of authoritative records field consists of the number of authoritative records in the authority portion of a reply message. Similarly, this field is filled by zeros for a query message. Finally, the number of additional records field records is in the additional information portion of a reply message and is similarly filled by zeros in a query message.

The questions field can contain one or more questions. The answers field belongs only to a reply message and consists of one or more replies from a DNS server to the corresponding client. The authority field is present only in reply messages and provides the domain name information about one or more authoritative servers. Finally, the additional information field is present only in reply messages and contains other information, such as the IP address of the authoritative server. This additional information helps a client further identify an answer to a question.

The next section explains how domains are added to DNS database. New information and names are included into a DNS database through a registrar. On a request for inclusion of a new domain name, a DNS registrar must verify the uniqueness of the name and then enter it into its database.

6.3 ELECTRONIC MAIL

Electronic mail is the method of transmitting of messages over electronic communication networks. The messages can be sent from the keyboard or electronic files stored on disk. Some e-mail systems are confined to a single computer network and some computers can send e-mails anywhere in the world.

Sent messages are stored in electronic mailbox until the recipient reads them. The term e-mail to both Internet e-mail system based on the Simple Mail Transfer Protocol (SMTP) and to Intranet systems, which allows users within one organization to e-mail each other. SMTP is limited to deliver simple text messages. Since the demand for various types of data, such as voice, images, and video clips are increased, a new electronic mail standard is builds on SMTP. That is Multi Purpose Internet Mail Extension (MIME).

6.3.1 Simple Mail Transfer Protocol (SMTP)

Simple Mail Transfer Protocol (SMTP) is Internet's standard host-to-host mail transport protocol, which operates over TCP, port 25. SMTP is a simple, text based protocol. It will not work well with binary files because this protocol started out as purely ASCII text based. Standards like MIME were introduced to encode binary files for transfer through SMTP. SMTP is used to send mail to the recipient's mailbox. There are many methods to access the e-mails, two methods are very popular and they are POP3 and IMAP. These protocols permits users to access their messages stored on a remote mail server.

If a user wants to send a message to someone, the sender-SMTP launches a two way transmission channel to a receiver-SMTP. SMTP commands are created by the sender-SMTP and sent to the receiver-SMTP. SMTP replies from the receiver-SMTP to the sender-SMTP in response to the commands. The message may be sent via one or more relay SMTP-servers, if direct connection does not exist between the sender and the final destination. The relay SMTP-servers acts as receivers and then sends the message to the next SMTP. SMTP-server must know the name of the destination host and the destination mailbox name to provide the relay.

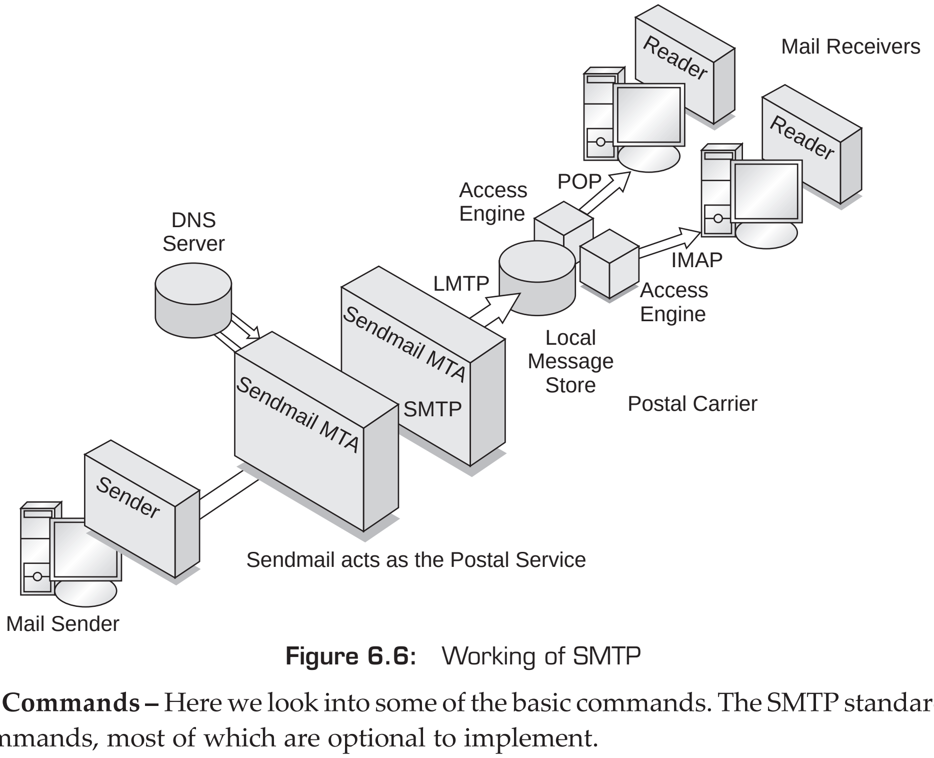


Figure 6.6: Working of SMTP

SMTP Commands – Here we look into some of the basic commands. The SMTP standard defines many commands, most of which are optional to implement.

(i) **HELLO (HELO)** — This is the first command that is sent when a connection is established. It is used to identify the sender-SMTP to the receiver-SMTP. The argument field contains the host name of the sender-SMTP.

HELO <SP><domain><CRLF>

<SP> stands for a space and <CRLF> stands for a combination of Carriage Return and Linefeed. The receiver-SMTP identifies itself to the sender-SMTP in the connection greeting reply, and in the response to this command.

(ii) **MAIL (MAIL)** — There are three steps to SMTP mail transactions. The transaction is started with a MAIL command which gives the sender identification. A series of one or more RCPT commands follows giving the receiver information. Then a DATA command gives the mail data. And finally, the end of mail data indicator confirms the transaction. The first step in the procedure is the MAIL command. The <reverse-path> contains the source mailbox.

MAIL <SP> FROM :<reverse-path> <CRLF>

If accepted, the receiver-SMTP returns a 250 OK reply. The <reverse-path> can contain more than just a mailbox. The <reverse-path> is a reverse source routing list of hosts and source mailbox. The first host in the <reverse-path> should be the host sending this command.

(iii) **RECIPIENT (RCPT)** — This command gives a forward-path identifying one recipient. If accepted, the receiver-SMTP returns a 250 OK reply, and stores the forward-path. If the recipient is unknown the receiver-SMTP returns a 550 Failure reply. This second step of the procedure can be repeated any number of times.

RCPT <SP> TO: <forward-path><CRLF>

The <forward-path> can contain more than just a mailbox. The <forward-path> is a source routing list of hosts and the destination mailbox. The first host in the <forward-path> should be the host receiving this command.

(iv) **DATA (DATA)** — This is the third step in the procedure, which is called DATA command.

DATA <CRLF>

If accepted, the receiver-SMTP returns a 354 Intermediate reply and considers all succeeding lines to be the message text. When the end of text is received and stored the SMTP-receiver sends a 250 OK reply. SMTP indicates the end of the mail data by sending a line containing only a period. The mail data includes the memo header items such as Date, Subject, To, Cc, From etc.

(v) **VERIFY (VRFY)** — This command asks the receiver to confirm that the argument identifies a user.

VRFY <SP><user name><CRLF>

(vi) **RESET (RSET)** — This command specifies that the current mail transaction is to be aborted. The receiver must send an OK reply.

RSET <CRLF>

(vii) **NOOP (NOOP)** — This command does not affect any parameters or previously entered commands. It specifies no action other than that the receiver send an OK reply.

NOOP <CRLF>

(viii) **QUIT (QUIT)** — This command specifies that the receiver must send an OK reply, and then close the transmission channel.

QUIT <CRLF>

The following minimum implementation is required for all receivers to make SMTP workable,

- MAIL
- RCPT
- DATA
- RSET
- NOOP
- QUIT

6.4 MULTIPURPOSE INTERNET MAIL EXTENSION (MIME)

Initially e-mail consisted of text messages written in English and expressed in ASCII. But now the demand is more such as messages with audio and video, messages in languages without alphabets, for example; Chinese and Japanese. On the Internet, data is sent as 8 bit bytes. The receiving software collects these bytes and assembles them in the correct order.

In addition to the bytes, few extra bytes are sent to convey what the data is. This job is done by MIME. It informs what is in a message so that the message contents can be used in an appropriate way. Basically all human-written Internet e-mail and automated e-mail are transmitted via SMTP in MIME format.

MIME can send other kind of information in e-mail, such as text in other languages using character encoding other than ASCII as well as 8-bit binary content like images, sounds, movies, and computer programs. MIME is a fundamental component of communication protocols such as HTTP. HTTP needs that data be transmitted in the context of e-mail like messages, though the data may not be e-mail.

The RFC 2822 defines the basic format of Internet e-mail and this is an updated version of RFC 822. These standards explain different formats for text e-mail headers and body. These rules relevant to header fields such 'To', 'Subject', 'From', and 'Data'. MIME explains rules for encoding non-ASCII characters in e-mail message header, such as 'Subject' and it permits header fields to contain non-English characters.

The basic structure of a MIME consists of a list of headers and a body. The headers describe the structure and content of the part and the header describes the part body as plain text in the US-ASCII character set. MIME headers don't have anything to do with the transport protocol used for the message. The body of a MIME is the content of the part. The content can be simple data, such as unstructured text or an encoded image.

MIME Headers

MIME defines a number of new RFC 822 header fields that are used to describe the content of a MIME entity. Each MIME header contains information about the part. The most important aspect of using the MIME format is to understand the MIME headers. The following section explains the header supported by the MIME package.

- **MIME-Version –** A MIME message should have a header with the information of the version of the MIME format, which the message follows. The name of this header is MIME-Version. The MIME-Version header specifies the version of MIME used to construct the MIME part. The

MIME package supports MIME version 1.0, so the value of this header is 1.0. The header is required only for a top-level MIME message.

Example: MIME-Version: 1.0

- **Content-Type–** The content-Type header explains the type of content the message has.

  Example 1: Content-Type: image/jpeg

  Example 2: Content-Type: text/plain

  The value of the header has two parts, separated by a forward slash. The first part of the value is the *media type*, a general description of the content. The second part of the value is the *subtype*, the specific type of the content. The MIME specification defines a multipart content type for a part body that contains other MIME parts. All *multipart*content types require a boundary parameter. The boundary is used to restrict the other MIME parts contained within the part body. Because the boundary is used as a delimiter, and the boundary should not appear in any of the contained parts.

  Example: Content-Type: multipart/mixed; boundary="_bounds_"

  If a part does not have a Content-Type header, the content type of the part is text/plain; charset=us-ascii.

- **Content–Transfer–Encoding** – The Content-Transfer-Encoding header describes whether the body has been processed for transmission. The header also describes the current representation of the body. The field name of this header is Content-Transfer-Encoding. The MIME specification defines five distinct transfer encodings. Two of these encodings, *base64* and *quoted-printable*, indicate that the part body was processed for transmission, and is currently represented as 7-bit ASCII. The remaining three encodings, *7bit, 8bit,* and *binary,* indicate that the body has not been processed for transmission, and that the message is currently represented as either 7-bit ASCII, 8-bit ASCII, or as raw binary data.

- **Content-Description** – The Content-Description header briefly describes the body. This header is only a human-readable description. It does not affect the structure or content of the part.
  Example: content-Description: very simple MIME message

- **Content-ID** – The content-ID header relates a unique ID with a MIME part. A Content-ID header is designed to permit external references to given MIME part. Hence this ID must be world unique. Other parts can refer to the part by using the identifier, even if the other parts are in a different message.

  Example: Content-ID: <part91034@roguewave.example>

- **Content-Location** – The Content-Location header associates a URI with the part. Other parts in the same message can refer to the part body by using the URI.

  Example: Content-Location: http://roguewave.example/lake.jpg

- **Content-Disposition** – The Content-Disposition header indicates how an application should process a part.

  Example: content-Disposition: inline

Part Body

The body of a part is the actual content of the part. The body can be simply data or, if the Content-Type of the message is multipart, the body can contain other MIME parts.

6.5 INTERNET DIRECTORY SERVICE AND WORLD WIDE WEB

The Directory Services play an important role in the development and support of the e-Business infrastructure of any enterprise. Organizations are finding that the applications often share similar requirements for

- Identity management
- Security and authentication
- Access controls
- Personalization
- Provisioning

The growing number of applications in the enterprise often means that each that each application devices it own mechanism for dealing with each of the above areas. This is costly and confusing since it means that the same information is often maintained in different places, in different ways and with incompatible technologies.

The most effective way to make information management more effective within organizations is to integrate different applications into the use of a single underlying directory service or a tightly integrated set of directory services.

SUMMARY

- The application layer enables the user or software to access the network.
- Electronic mail is the method of transmitting of messages over electronic communication networks.
- Simple Mail Transfer Protocol (SMTP) is Internet's standard host-to-host mail transport protocol, which operates over TCP, port 25.
- The SMTP standard defines many commands, most of which are optional to implement.
- MIME can send other kind of information in e-mail, such as text in other languages using character encoding other than ASCII as well as 8-bit binary content like images, sounds, movies, and computer programs.
- MIME defines a number of new RFC 822 header fields that are used to describe the content of a MIME entity.
- The body of a part is the actual content of the part.
- DNS is used mostly to translate between domain names and IP addresses, and to control email delivery.
- The URL is a standard for specifying any kind of information on the Internet.
- Uniform Resource Identifier is the unique name used to access the resource.
- The Hypertext Transfer Protocol (HTTP) is used to access data on the World Wide Web.

Section-II

Advanced Computer Networks

7

WIDE AREA NETWORKS

Introduction

A WAN is a data communications network that covers a relatively broad geographic area, often a country or continent and, and use communications circuits to connect the intermediate nodes. It often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer. Figure 7.1 illustrates the relationship between the common WAN technologies and the OSI model.

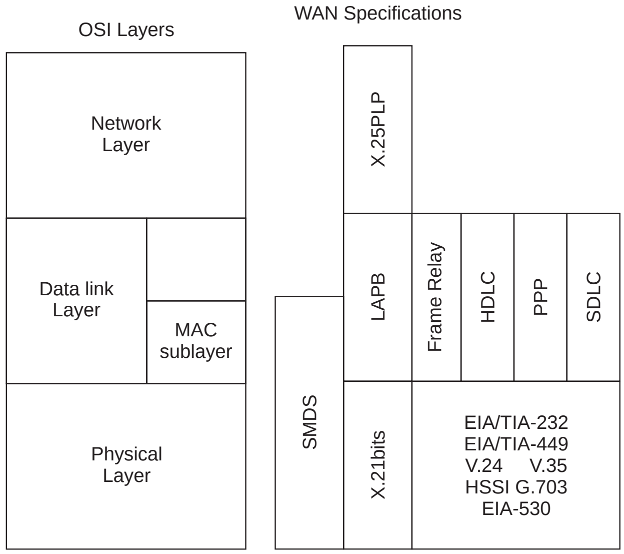


Figure 7.1: Relationship between WAN technologies and OSI model

WAN technologies function at the lower three layers of the OSI reference model (physical, data link, and the network layers). The above figure shows the levels at which WAN technologies like Frame Relay, ATM, etc. operate at.

7.1 FRAME RELAY

Frame Relay was developed in the late 1980s and early 1990s as a virtual-circuit WAN in order to meet the demand for a new type of WAN. This high performance WAN protocol functions at the physical and data link layers of the OSI model. Although it was originally developed to work across ISDN interfaces, it can be used over other network interfaces also.

Prior to frame relay X.25 was used as the virtual-circuit switching network. Disadvantages associated with X.25 are its low 64 Kbps data rate, extensive flow and error control at both the network and data link layers, and high overhead as X.25 has its own network layer.

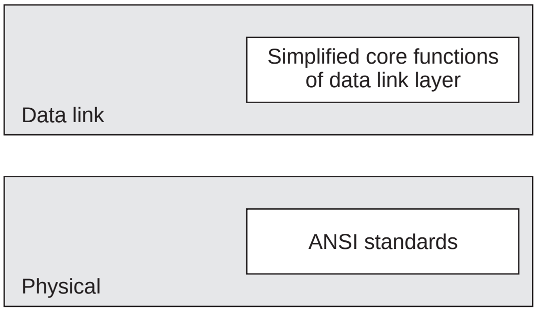


Figure 7.2: Frame relay protocol

Frame Relay has the following features that overcome the drawbacks of X.25:

- High operating data speed of 1.544 Mbps and even as high as 44.376 Mbps.
- Since it functions at the physical and data link layers, frame relay can be used as a backbone network to those protocols that already have a network protocol like the Internet.

  Supports *bursty* traffic.
- Frame size is 9000 bytes, thus being able to accommodate all other LAN frame sizes.
- Less expensive that other WANs.
- Error detection at the data link layer only.

The technology used in frame relay is packet switching. Using this, end terminals can share the network medium and bandwidth dynamically. Two important techniques used in packet switching technology are:

- ***Variable-length packets***-The packet size can be variable which warrants a more efficient and flexible data transfer.
- ***Statistical multiplexing***-This technique monitors network access in packet switched networks leading to a more efficient bandwidth usage. Since most of the LANs in use these days like the Ethernet and Token ring are packet switched networks, frame relay will prove to be more all the more useful.

7.1.1 Frame Relay Devices

The devices in a Frame Relay Wide Area Network are of generally two types:

- Data Terminal Equipment (DTE)
- Date Circuit-terminating Equipment (DCE)

DTEs are generally the computers, routers, bridges, and terminals seen on the customer premises. These devices usually are the terminating equipment in a network and are mostly owned by the customer.

On the other hand, DCEs are owned by the carriers. These devices are responsible for transmitting data through the WAN and are the switching/clocking devices in the WAN. E.g. packet switches. The following figure shows DTEs and DCEs in a typical network.

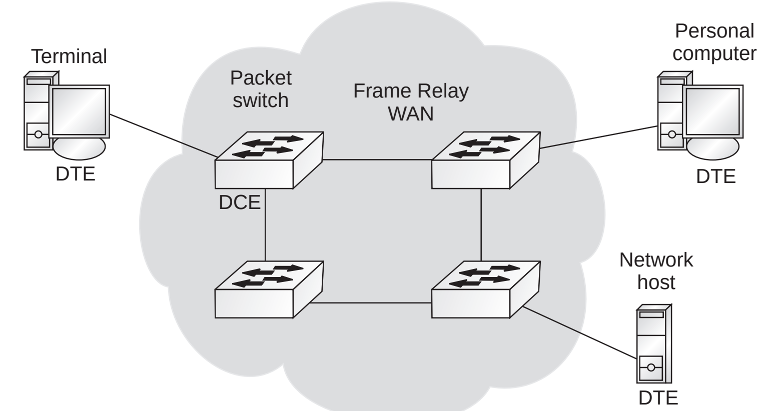


Figure 7.3: Arrangement of DTEs and DCEs in a WAN

Since the connection between DTEs and DCEs are at both the physical and data link layer levels, specifications are required for both the physical layer interface and the data link layer component. One of the most commonly used physical layer interface specifications is the recommended standard (RS)-232 specifications. The protocol that defines the connection between DTE devices and DCE devices can be a common WAN protocol like Frame Relay.

7.1.2 Virtual Circuits

Frame relay is connection-oriented; thus a defined connection will exist between devices. A frame relay virtual circuit implements this concept with a logical connection between DTE devices. The data link connection identifier (DLCI) is used as a virtual circuit identifier (VCI) in frame relay to identify the specific virtual circuit. Multiple virtual circuits are multiplexed onto a single link so that data transmission can occur over the network.

The following figure shows a typical frame relay network which is connected to the Internet. Routers are used to interconnect LANs and WANs in the Internet.

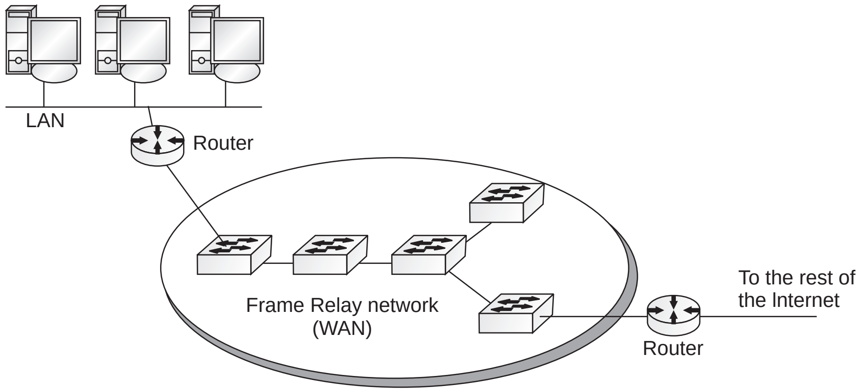


Figure 7.4: Frame relay virtual circuit

Frame Relay virtual circuits fall into two categories: switched virtual circuits (SVCs) and permanent virtual circuits (PVCs).

7.1.2.1 Switched Virtual Circuits (SVC)

SVCs are temporary, short connections that exist only in those situations when data is to be transferred between source and destination, i.e., only sporadic data transfer. The communication session across an SVC has four states:

- Call setup: Establishes the virtual circuit between the DTE devices.
- Data Transfer: Actual transmission of data over the virtual circuit.
- Idle: In this state, the connection is still active but there is no transfer of data. If this state occurs for more than a specific period of time, the call gets terminated.
- Call Termination: The virtual circuit is terminated.

If more data has to be exchanged or transmitted, a new SVC must be established. SVCs have become quite popular these days as organizations have realized that SVCs are cost-saving as the circuit is not *open* all the time.

7.1.2.2 Permanent Virtual Circuits (PVC)

As the name suggests, PVCs are permanent connections between DTE devices and is mostly set up when there are chances of frequent data exchanges between devices across a Frame Relay network. The connection setup is simple with corresponding table entries being made at all switches by the administrator. The source is given an outgoing DLCI while the destination is assigned an incoming DLCI.

The communication sessions across a PVC do not need a call setup and call termination phase as seen in SVCs. It has only two states:

- Data Transfer
- Idle: In the idle state, the connection is active but no data transfer takes place. However, the PVC is not terminated unlike in the case of SVCs.

The advantage with PVCs is that devices can start exchanging data anytime as the circuit is permanent. The drawbacks are:

- o PVCs are expensive as both sides have to pay for the connection even if it is not used.
- o A separate PVC is needed for every connection.

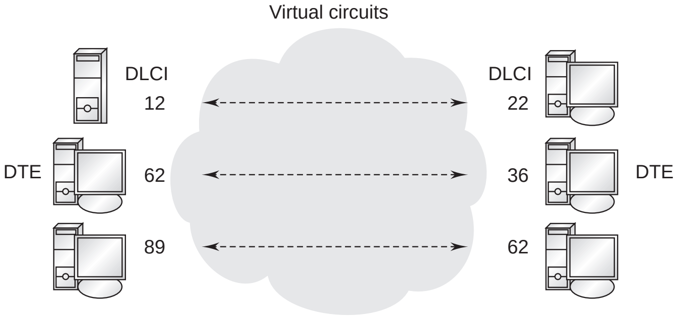


Figure 7.5: Permanent virtual circuit

7.1.3 Data-Link Connection Identifier (DLCI)

As discussed before, a frame relay virtual circuit is uniquely identified by the data-link connection identifier (DLCI). These values are generally allocated by the frame relay service provider. The DLCI is unique only within the LAN, it may not be unique in the Frame Relay WAN. The following figure shows how the same DLCI value (62) has been assigned to two devices in the same WAN (The cloud depicts the frame relay network).

7.1.4 Frame Relay Network

When implementing a private Frame Relay network, ensure that the T1 multiplexer has/ or is equipped with both Frame Relay and non-Frame Relay interfaces. In this way, the frame relay traffic can be sent out via the Frame Relay interface onto the network. Similarly, non-frame relay traffic can be sent to the related service (A PBX for telephone service or onto a video-teleconferencing application).

A typical Frame Relay network is shown in the following figure. It consists of several DTE devices like routers that are connected to remote ports on multiplexer equipment via traditional point-to-point services such as T1, fractional T1, or 56-Kb circuits.

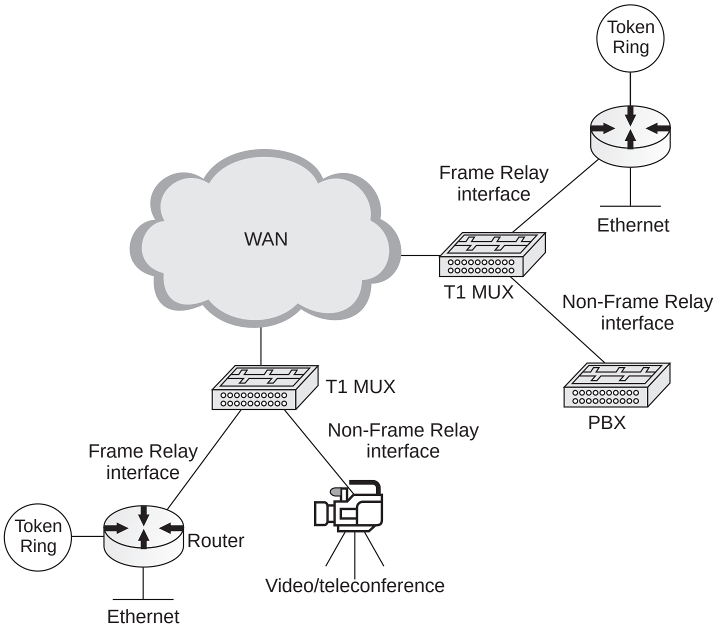


Figure 7.6: Frame relay network

7.1.5 Frame Relay Network Implementation

Frame relay networks are implemented as either ***public carrier-provided networks*** or ***private enterprise networks.***

7.1.5.1 Public Carrier-Provided Networks

Most of the Frame relay networks seen these days are public carrier-provided networks. In these types of networks, the switching equipment will be situated in the central offices (CO) of the public carrier. Based on their usage, customers are charged a fee. The DCE equipment is the property of the telecommunications provider whereas the DTE equipment could be customer owned or given to the customer by the provider. It is the telecommunications provider that is responsible for the administration and maintenance of the Frame Relay equipment.

7.1.5.2 Private Enterprise Networks

The difference seen in private enterprise networks when compared to public carrier-provided networks is that the management of the network is the sole responsibility of the enterprise (a private company). The equipment (both DTE and switching equipment) is the property of the customer.

7.1.6 Standard Frame Relay Frame

A standard frame relay frame is shown in the following figure:

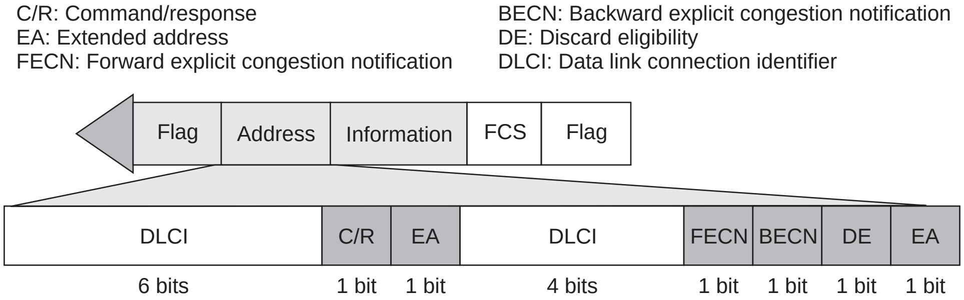


Figure 7.7: Frame relay frame format

The various fields are described as follows:

- **Flags:** Indicates the beginning and end of a frame relay frame. The value is represented as the hexadecimal number 7E or the binary number 01111110.
- **Address:** Has several fields which are:
  - o **DLCI (10 bits):** Gives the data link connection identifier for the virtual connection. The first 6 bits in the first byte is the first part of the DLCI. The first four bits of the second byte make up the second part of the DLCI. The DLCI value is significant only to the physical channel the circuit resides on.
  - o **C/R Command/Response(1 bit):** Allows upper layers to realize if the frame is a command or a response. This bit is not used by the frame relay protocol.
  - o **EA Extended Address (1 bit):** Indicates whether the current byte is the final byte of the address. If EA=0, then another address byte is to follow. If EA=1, then the current byte is the last DLCI octet. The eighth bit of each byte of the address field indicates the EA.
  - o **FECN (Forward Explicit congestion notification- 1 bit):** If this bit is set, then it indicates to a DTE device (destination) that congestion has occurred in the direction of the frame transmission from source to destination. The destination will thus understand that there will be a delay in the reception of packets.
  - o **BECN (Backward explicit congestion notification – 1 bit):** If set, this bitlets the sender know that congestion has occurred in the direction opposite of frame transmission from source to destination. Thus the sender will know that it should reduce the speed of sending packets so that packet loss is minimized.
  - o **DE (Discard Eligibility – 1 bit):** This bit, if set, lets the network known that this frame can be discarded if there is any congestion in the network. It sets the priority of the frame with respect to other frames in the network. It can be set by the sender of the frame or any switch in the network.

- **Information/Data:** Can vary in length upto 16,000 octets.
- **FCS (Frame Check Sequence):** This value is calculated by the sender and is verified by the destination so that the transmission integrity is ensured.

7.2 ASYNCHRONOUS TRANSFER MODE (ATM)

Asynchronous Transfer Mode (ATM) is a network technology standard where data is transferred in fixed-size cells. The benefit of a fixed size and fixed format is an efficient transmission scheme over high-speed networks. ATM scores over LAN and WAN technologies in the areas of bandwidth and Quality of Service (QoS). This makes it easier for newer applications like multimedia to be used as well. In addition, the small cell size allows the transmission of various types of information like video, audio, and computer data over the same network and at the same time, ensure that no one data type takes up the line. The following topics summarize ATM technology, operations and logical connections.

7.2.1 Definition and Overview

The streamlined packet transfer interface, which best describes ATM, employs fixed-size packets known as ***cells*** for data transfer. Information in multiple types like voice, video, or data can be conveyed in these small cells. ATM (or Asynchronous Transfer Mode) is an International Telecommunication Union- Telecommunications Standards Section (ITU-T) standard and is connection-oriented. The following figure shows a public and a private ATM network capable of carrying voice, data, and video traffic.

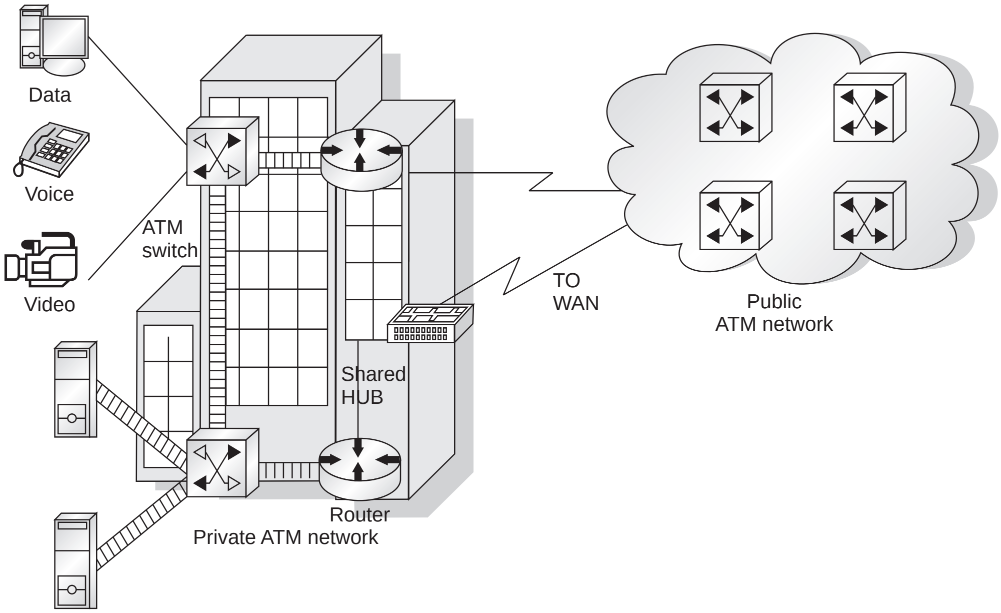


Figure 7.8: Data transfer using ATM

From a technical point of view, the Asynchronous Transfer Mode technology is an evolution of packet switching. It was developed as a result of work on broadband ISDN. It can be applied even in non-ISDN areas, which require very high data rates. ATM combines the multiplexing and switching functions, which makes it suitable for bursty traffic (not supported by circuit switching) and for communication between devices operating at various speeds.

The ATM technology is very similar to packet switching using X.25 and to even frame relay as the data being transferred is in the form of chunks of data. Another similarity between ATM, packet switching, and frame relay is how multiple logical connections can be multiplexed over a single physical interface. But unlike packet switching, ATM supports high-performance multimedia networking. Some of the networking devices that have ATM implemented on them are:

- PCs, workstations, and server network interface cards
- Switched-Ethernet and token-ring workgroup hubs
- Workgroup and campus ATM switches
- ATM enterprise network switches
- ATM multiplexers
- ATM–edge switches
- ATM–backbone switches

The ATM technology will decrease the cost involved in infrastructure by its efficient bandwidth management, simple operation, and the merging of overlay networks. The money and time involved in setting up a separate network for every new service requirement like frame relay is something that carriers are not willing to take up these days. The ATM technology ensures that there is a core network stability but at the same time ensures that other service interfaces can evolve or grow at a rapid pace.

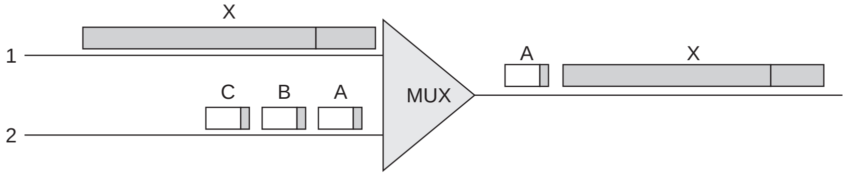
7.2.2 Benefits of ATM

ATM came up as the solution to the ever-increasing demand for quality of service on networks that support various types of information like data, video, voice, and audio. The following benefits are seen on networks that have ATM services deployed on them:

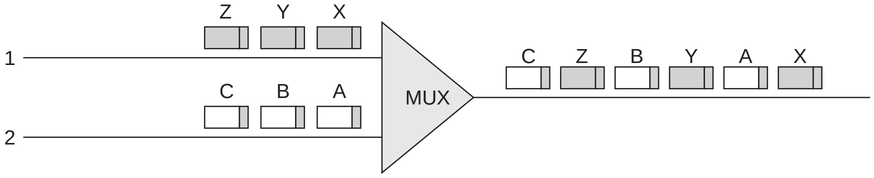
1. High speed communication
2. High performance with hardware-based switching.
3. Flexible and dynamic allocation of bandwidth for bursty traffic. The efficient allocation of bandwidth seen in ATM utilizes networking resources to their maximum. The LAN-based data applications, voice, and video are all bursty, therefore ATM is the best answer for a network connection that can be reliably mix voice, data, and video.
4. Multimedia traffic, in general, has different degrees of throughput and latency. ATM provides the best support for multimedia applications on a single network.
5. ATM provides a scalability in speed and network size that can support speeds from T1/E1 to OC-12, which is upto 622 Mbps. This will encompass the multi-Gps range very soon.
6. Has a connection-oriented technology, which is similar to the traditional telephony.
7. The LAN/WAN architecture is common for ATM, which allows its consistency from computer to computer.
8. Opportunities for simplification via switched VC architecture.
9. A single, universal, interoperable network transport.

7.2.3 ATM Technology

Previous to the ATM networks, data was communicated via frame switching and frame networks. The different protocols at the various layers all used various sized frames. As and when the networks got more complex, the information that needed to be carried in the header got more complex and large in size when compared to the data that was to be sent. To improve efficiency, even if the data unit was made larger, it would be a waste if there were little data to be sent as the data unit field would not fully utilized. This caused protocols to allow variable frame size based on the data unit size.



The variable frame sized network would now have a mixed network traffic. Networking between the routers, switches etc with variable frame sizes made it necessary to have complex software to manage and monitor these frames. Also, the constant uncertainty in frame size made it impossible to guarantee a data rate delivery. With time division multiplexing, imagine two frames (one particularly huge) being multiplexed onto one single link.



If the longer frame arrives at the multiplexer a second earlier than the smaller frames, it is sent onto the link causing the small frame to wait for a long time before it can get transmitted. So, if audio and video frames, that are usually quite small, are mixed with ordinary data frames, they will face lots of delays. In this way, users would not want to send their audio or video frames over any shared link.

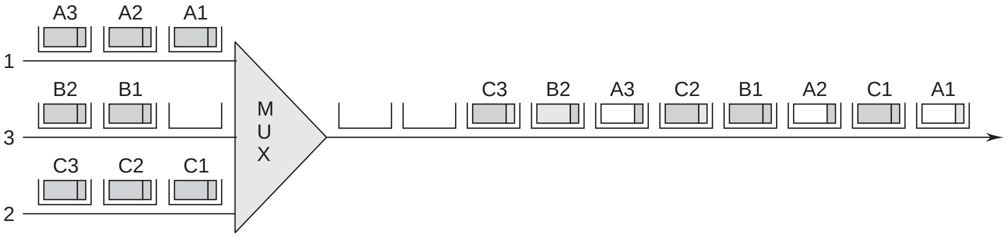


Figure 7.9: Multiplexing in ATM

The solution to this was to adopt cell networking where a cell refers to a small data unit of a fixed size. These cells would be the basic unit of data exchange with all data being loaded into identical cells which can transmitted uniformly. When different sized data frames from various types of networks reach a cell network, they are split up into data units of the same size and put into cells. These cells are then multiplexed onto shared links. Since all cells are of the same size, there will be no delays or wastage of frame space. The following figure shows how multiplexing would take place with fixed-size cells:

The interleaving between the two lines ensures that no cell undergoes a long delay at the multiplexer. Also, the cells reach their destination in a continuous stream thus proving that cell networks can manage real-time transmissions.

ATM employs **asynchronous time division multiplexing** (that is why this technology is known as asynchronous transfer mode) when multiplexing cells from different networks. The time slots, which are of a fixed size as well, are loaded with cells from channels. If there are no cells, that slot would be empty.

The Figure 7.9 shows cells coming in from three different channels are multiplexed onto a single link. Since channel 2 has no cell to transmit at the first clock tick, the multiplexer takes the cell from channel 3 to fill in the time slot. Finally, when all the cells have been multiplexed onto the link, the output slots are shown as being empty.

7.2.4 ATM Architecture

The typical ATM network consists of:

- User-to-Network Interface (UNI)
- Network-to-Network Interface (NNI)
- Endpoints
- Switches

The *endpoints*, which are the user access devices, are connected to switches inside the network using the *User to Network Interface or UNI.* Within the network, the switches are connected to each other via the *Network-to-Network Interface or NNI.*

A typical ATM network is shown in the following figure:

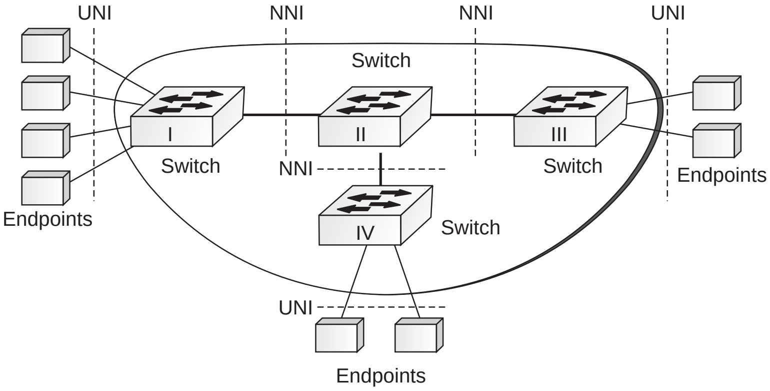


Figure 7.10: ATM architecture

The end points in an ATM network are connected via ***Transmission paths (TP), Virtual Paths (VP), and Virtual Circuits (VC).*** Transmission paths refer to the actual physical media between the switches or between the endpoints and a switch. It can be compared to the collection of all highways connecting the cities of Mumbai and Pune.

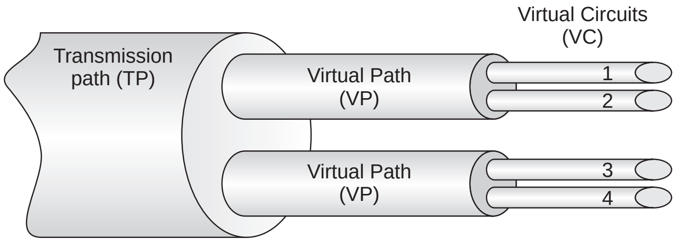


Figure 7.11(a): Physical media for an ATM network

The Transmission paths have several virtual paths within them. A virtual path (VP) is the connection between two switches. Taking up the comparison of highways connecting two cities, then a virtual path would refer to a single highway between Mumbai and Pune.

Virtual circuits (VC) refer to the *path* that cells of a single message will follow to reach its destination. A virtual circuit can be compared to the lanes in a highway from Mumbai to Pune.

The ATM developers designed hierarchical identifiers for the virtual connections that data would take when being transmitted from one endpoint to the other. The identifier consists of: ***a Virtual Path Identifier (VPI)*** and a ***Virtual Circuit Identifier (VCI).*** The VPI will specify the VP while the VCI will identify which VC within the VP is being used. The following figure shows how a virtual connection is identified by the pair of VPI and VCI.

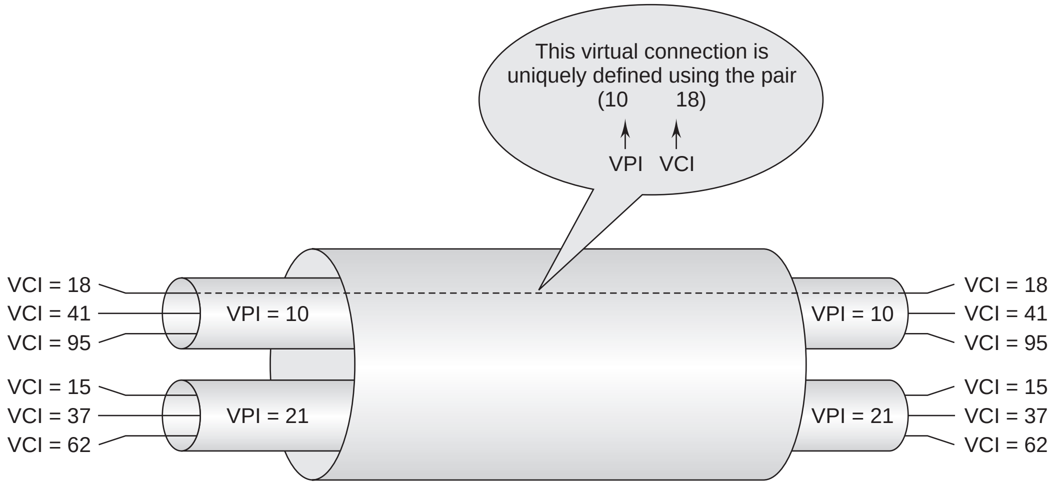


Figure 7.11(b): Identifiers for virtual connection

The ATM cell is of a fixed-length of ***53 bytes*** (or octets) where 48 bytes is the payload while 5 bytes make up the cell header. Having a fixed cell size ensures that data that are time-sensitive like voice or video are not delayed or affected by large sized packets. The header carries information on the type of payload, VCI, and header error check.

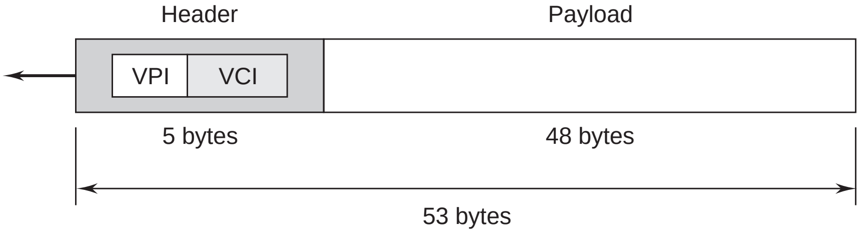


Figure 7.12(a): ATM cell format

A detailed figurative view of the ATM cell header is shown in the following figure:

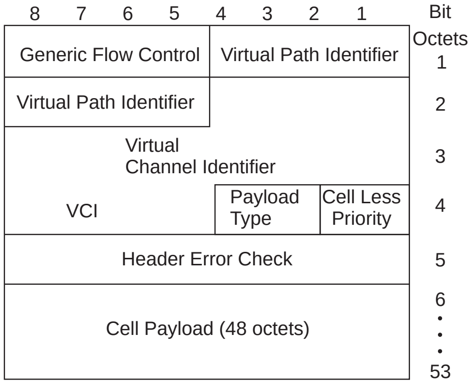


Figure 7.12(b): ATM cell header

7.2.5 ATM Classes of Services

The ATM Forum UNI 4.0 has specified five classes of service for ATM. They are:

1. Constant Bit Rate (CBR)
2. Variable Bit rate-non-real time (VBR-NRT)
3. Variable Bit Rate-Real Time (VBR-RT)
4. Available bit Rate (ABR)
5. Unspecified Bit Rate (UBR)

Table 7.1: ATM service classes

Service Class	**Quality of Service Parameter**
Constant Bit Rate (CBR)	This class is used to emulate circuit switching. The cell rate is constant with time. CBR applications are quite sensitive to cell-delay variation. Examples of applications that can use CBR are telephone traffic (i.e., nx64 kbps), videoconferencing, and television.
Variable Bit rate–non-real time (VBR–NRT)	Allows users to send traffic at a rate that varies with time depending on the availability of user information. Statistical multiplexing is provided to make optimum use of network resources. Multimedia e-mail is an example of VBR–NRT.
Variable bit rate–real time (VBR–RT)	Is similar to VBR–NRT but is designed for applications that are sensitive to cell-delay variation. E.g: voice with speech activity detection (SAD) and interactive compressed video.
Available bit rate (ABR)	Provides rate-based flow control and is aimed at data traffic such as file transfer and e-mail. Although the standard does not require the cell transfer delay and cell-loss ratio to be guaranteed or minimized, it is desirable for switches to minimize delay and loss as much as possible. Depending upon the state of congestion in the network, the source is required to control its rate. The users are allowed to declare a minimum cell rate, which is guaranteed to the connection by the network.
Unspecified bit rate (UBR)	This class is the catch-all, other class and is widely used today for TCP/IP.

The following table describes the technical parameters that have been identified by the ATM forum and will be associated with an ATM connection.

Table 7.2(a): Technical parameters for ATM

Technical Parameter	**Definition**
Cell loss ratio **(CLR)**	CLR is the percentage of cells not delivered at their destination because they were lost in the network due to congestion and buffer overflow.
Cell transfer delay **(CTD)**	The delay experienced by a cell between network entry and exit points is called the CTD. It includes propagation delays, queuing delays at various intermediate switches, and service times at queuing points.
Cell delay variation **(CDV)**	CDV is a measure of the variance of the cell transfer delay. High variation implies larger buffering for delay-sensitive traffic such as voice and video.
Peak cell rate **(PCR)**	The maximum cell rate at which the user will transmit. PCR is the inverse of the minimum cell inter-arrival time.
Sustained cell rate **(SCR)**	This is the average rate, as measured over a long interval, in the order of the connection lifetime.
Burst tolerance **(BT @ PCR)**	This parameter determines the maximum burst that can be sent at the peak rate. This is the bucket-size parameter for the enforcement algorithm that is used to control the traffic entering the network.

These parameters are present only in some of the ATM classes of service:

Table 7.2(b): Presence of parameters in ATM service classes

Class of Service	**CBR**	**VBR–NRT**	**VBR–RT**	**ABR**	**UBR**
CLR	✓	✓	✓	✓	✗
CTD	✓	✗	✓	✗	✗
CDV	✓	✓	✓	✗	✗
PCR	✓	✓	✓	✗	✓
SCR	✗	✓	✓	✗	✗
BT @ PCR	✗	✓	✓	✗	✗
flow control	✗	✗	✗	✓	✗

7.3 SONET

Short for Synchronous Optical Network, a standard for connecting fiber-optic transmission systems. SONET was proposed by Bellcore in the middle 1980s and is now an ANSI standard.

SONET defines interface standards at the physical layer of the OSI seven-layer model. The standard defines a hierarchy of interface rates that allow data streams at different rates to be multiplexed.

The international equivalent of SONET, standardized by the ITU, is called SDH.

7.3.1 Basic Concepts

The high bandwidth achievable by fiber-optic cables makes it suitable for the high data rate technologies like videoconferencing that are prominent these days. The fiber optic cables are capable of carrying large numbers of low rate technologies as well. ***Synchronous optical network (SONET)*** is a standard for optical telecommunications transport formulated by the Exchange Carriers Standards Association (ECSA) for the American National Standards Institute (ANSI). If SONET is the ANSI standard for fiber optic networks, then the ITU-T standard is known as ***Synchronous Digital Hierarchy (SDH).***

Advantages of SONET are:

- A reduction in equipment requirements and an increase in network reliability
- Overhead and payload bytes are provided wherein the overhead bytes allow the management of payload bytes individually thus assisting a centralized fault *sectionalization*.
- A synchronous multiplexing format to carry lower level digital signals was defined. In addition, the synchronous structure ensures that the interface to digital switches, digital cross-connect switches, and add-drop multiplexers is simple.
- A set of generic standards which allowed the connection of multi-vendor products is available.
- The architecture is designed to be flexible enough so that it can accommodate future applications, with a variety of transmission rates.

7.3.1.1 Signals

A hierarchy of electrical signaling levels referred to as synchronous transport signals (STSs) is defined by SONET. These STS levels vary between STS-1 to STS-192 and support various data rates (in Mbps). With respect to each of these STS levels, there is the corresponding optical signal, also known as optical carriers (OCs). SDH has its own signaling level and associated names system, known as synchronous transport module (STM). STM is compatible with European data rate hierarchies like E lines and also with STS levels.

Table 7.3: Signaling levels for SONET

STS	OC	Rate (Mbps)	STM
STS - 1	OC - 1	51.840	
STS - 3	OC - 3	155.520	STM - 1
STS - 9	OC - 9	466.560	STM - 3
STS - 12	OC - 12	622.080	STM - 4
STS - 18	OC - 18	933.120	STM - 6
STS - 24	OC - 24	1244.160	STM - 8
STS - 36	OC - 36	1866.230	STM - 12
STS - 48	OC - 48	2488.320	STM - 16
STS - 96	OC - 96	4976.640	STM - 32
STS - 192	OC - 192	9953.280	STM - 64

7.3.1.2 Multiplexing Scheme

The SONET technology employs a byte-interleaving multiplexing scheme by which multiple signals of different capacities can be carried through a synchronous and flexible optical hierarchy.

In the SONET multiplexing process, the base signal (synchronous transport signal–level 1, or simply STS–1) is generated first. The higher signals are just integer multiples of STS-1. An STS–N signal is composed of N byte-interleaved STS–1 signals. Note the optical counterpart for each STS–N signal, with its designated optical carrier level N (OC–N) in the table. SONET topology can be a mesh, but most often it is a dual ring. Standard component of SONET ring is an ADM (Add/Drop Multiplexer).Drop one incoming multiplexed stream and replace it with another stream.

7.3.2 SONET Devices

To understand the SONET technology, take a look at a link that has SONET devices on it. These devices include:

1. STS multiplexers/de-multiplexers
2. Regenerator
3. Add/drop multiplexer
4. Terminal

The common transmission speeds in SONET is between 155 Mbps and 2.5 Gbps.

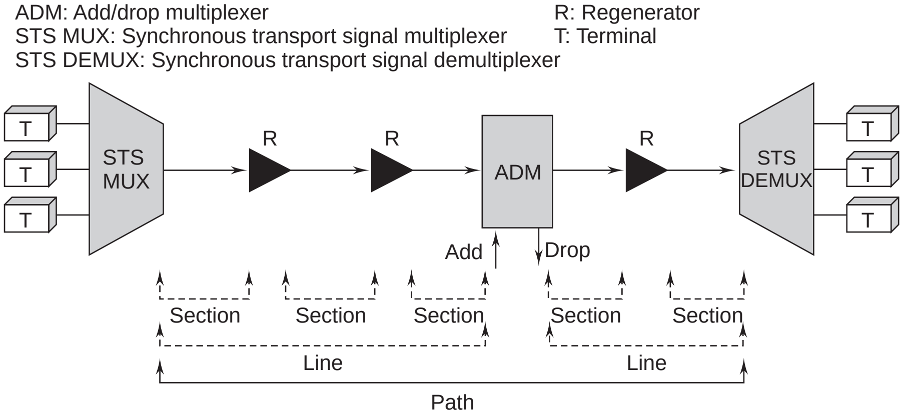

ADM: Add/drop multiplexer
STS MUX: Synchronous transport signal multiplexer
STS DEMUX: Synchronous transport signal demultiplexer
R: Regenerator
T: Terminal
T
T
T
STS MUX
R
R
ADM
R
STS DEMUX
T
T
T
Add
Drop
Section
Section
Section
Section
Section
Line
Line
Path

Figure 7.13: Transmission process in SONET

7.3.3 SONET Multiplexing

The basic concept in SONET is the ***multiplexing*** of channels (each of bandwidth as little as 64 Kbps) into ***data frames*** that are sent at fixed intervals. Ethernet cables can transmit data only upto 100 m or 328 feet. Comparatively, SONET can transmit data to much longer distances. Even *short reach* links span up to 2 kilometers (1.2 miles); *intermediate* and *long reach* links cover dozens of kilometers.

The ***terminals*** are devices that deploy the SONET network services. E.g. routers. The *STS **multiplexer/demultiplexer*** is the interface between the electrical circuit on one side and the optical network on the other side. The STS multiplexer will take up electrical signals from various sources and multiplexes it into an OC signal.

The STS demultiplexer, does the reverse. It demultiplexes the optical OC signal into the electrical signals. A *regenerator* is a repeater that lengthens the link by taking the optical signal (OC-n), demodulating it into an electrical signal (STS-n), regenerating it, and then modulating the regenerated signal back into an OC-n signal. The *Add/Drop multiplexer (ADM)* enables the addition or removal of signals.

It is possible that other STSs from other sources need to be added onto the signal. It is also possible that we do not want the entire signal to be demultiplexed. The ADM comes in handy at such times.

7.3.4 Benefits

One of the benefits of SONET is that it can carry large payloads (above 50 Mb/s). However, the existing digital hierarchy signals can be accommodated as well, thus protecting investments in current equipment. To achieve this capability, the STS Synchronous Payload Envelope can be sub-divided into smaller components or structures, known as Virtual Tributaries (VTs), for the purpose of transporting and switching payloads smaller than the STS-1 rate. All services below DS3 rate are transported in the VT structure.

Any type of service, ranging from voice to high-speed data and video, can be accepted by various types of service adapters. A service adapter maps the signal into the payload envelope of the STS-1 or virtual tributary (VT).

New services and signals can be transported by adding new service adapters at the edge of the SONET network. Except for concatenated signals, all inputs are eventually converted to a base format of a synchronous STS-1 signal (51.84 Mb/s or higher).

Lower speed inputs such as DS1s are first bit- or byte-multiplexed into virtual tributaries. Several synchronous STS-1s are then multiplexed together in either a single- or two-stage process to form an electrical STS-N signal (N = 1 or more).

STS multiplexing is performed at the Byte Interleave Synchronous Multiplexer. Basically, the bytes are interleaved together in a format such that the low-speed signals are visible. No additional signal processing occurs except a direct conversion from electrical to optical to form an OC-N signal.

7.3.5 Toplogy: SONET Rings

SONET supports the ring topology in which the building blocks are the Add/Drop Multiplxers (ADMs). The following figure shows how multiple ADMs can be arranged in a ring configuration to support either unidirectional/bidirectional traffic. The basic concept in a ring topology is the existence of two rings of fiber- the working ring and the protection ring.

Usually, only the working ring handles all the traffic. If the fiber cable is cut for some reason, the multiplexers have the intelligence to automatically detect the failure and transfer control to the protection ring in a matter of seconds thus offering uninterrupted data transmission. This is the reason why SONET is also referred to as a self-healing network technology.

This *automatic restoration* capability within seconds in addition to the demand for survivable services, diverse routing of fiber facilities, and flexibility to rearrange services to alternate serving nodes, have made rings a popular SONET topology.

7.3.6 Advantages

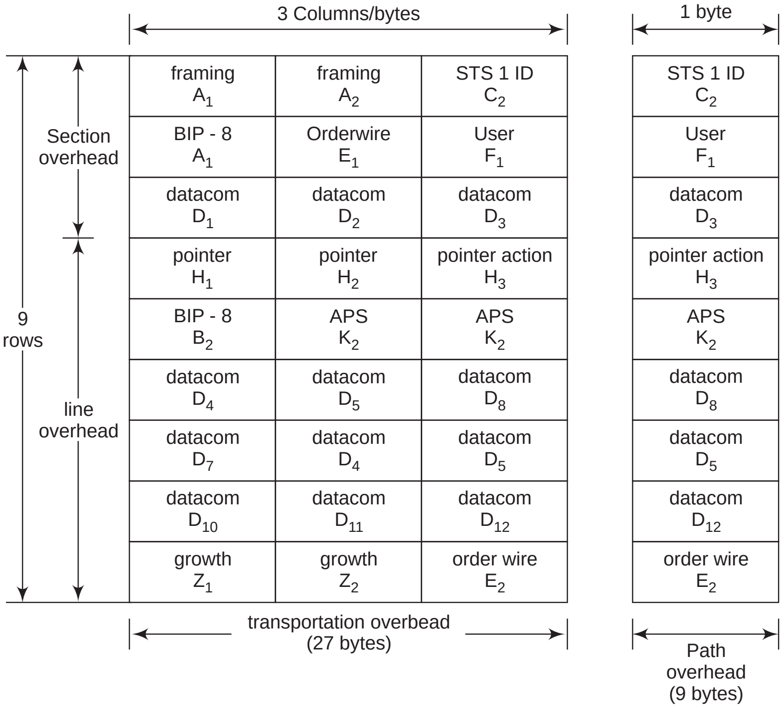
Advantages of having SONET as the network technology are:

1. Since SONET is based on direct synchronous multiplexing, slow signals can be multiplexed directly onto the higher speed SONET signals. There is no need of any in-between multiplexing stages.

2. Advanced network management and maintenance features.
3. Flexibility in accommodating any future networks that may need to be integrated.
4. Can be used in the three traditional telecommunications areas: long-haul networks, local networks and loop carriers. It can also be used to carry CATV video traffic.

7.3.7 SONET Frames

The basic SONET Frame structure is known as STS-1 or Synchronous Transport Signal level 1. It has a basic frame rate of 8,000 frames/sec = 51.84Mb/s. The frame length is 9-rows by 90-columns of octets — total frame size 810 octets. The first three columns are transport overhead, while columns 4-90 are the SPE - Synchronous Payload Envelope with actual transport speed 50.112Mb/s.



framing A_1	framing A_2	STS 1 ID C_2	STS 1 ID C_2
BIP - 8 A_1	Orderwire E_1	User F_1	User F_1
datacom D_1	datacom D_2	datacom D_3	datacom D_3
pointer H_1	pointer H_2	pointer action H_3	pointer action H_3
BIP - 8 B_2	APS K_2	APS K_2	APS K_2
datacom D_4	datacom D_5	datacom D_8	datacom D_8
datacom D_7	datacom D_4	datacom D_5	datacom D_5
datacom D_{10}	datacom D_{11}	datacom D_{12}	datacom D_{12}
growth Z_1	growth Z_2	order wire E_2	order wire E_2

Figure 7.14: SONET frame

SUMMARY

- **X.25** is a set of protocols developed by the CCITT/ITU which specifies how to connect computer devices over an internetwork.
- Frame relay networks provide a high-speed connection up to 1.544Mbps using variable-length packet-switching over digital fiber-optic media.
- However, Frame Relay does not provide flow or error control; they must be provided by the upper-layer protocols.
- A cell network uses the cell as the basic unit of data exchange. A cell is defined as a small, fixed-size block of information.
- A virtual connection is defined by a pair of numbers: the VPI and the VCI.

- Asynchronous Transfer Mode (ATM) is the cell relay protocol designed by the ATM Forum and adopted by the ITU-T.
- **Asynchronous Transfer Mode (ATM) m**ay be used over a variety of media with both baseband and broadband systems. It is used for audio, video, and data transfer. It uses fixed length data packets of 53 8 bit bytes called cell switching.
- **Synchronous Optical Network (SONET)** is a physical layer standard that defines voice, data, and video delivery methods over fiber optic media.
- SONET and SDH are based on circuit mode communication, meaning that each connection achieves a constant bit rate and delay.
- SONET supports the ring topology in which the building blocks are the Add/Drop Multiplexers (ADMs).
- SONET defines the optical carrier (OC) levels and electrically equivalent synchronous transport signals (STSs) for the fiber-optic–based transmission hierarchy

8

IPV6

Introduction

Address-depletion is a major concern for the Internet. IPv4 is still widely used as the internet protocol but experts believe that the four billion addresses that could be allocated using the 32-bit IPv4 address format are all but exhausted now. This situation was happening because government agencies are being allocated multiple large blocks of addresses. Thus, IPv6 came into the scene with an expansion in addressing from 32-bit (4octets) addresses to 128-bit (16 octets) addresses. This could be the answer to all the prayers for more addresses, but it has not become a standard as yet.

The IPv4 protocol could not support real-time audio and video transmissions, and encryption and authentication of data in some applications as well. Another much advanced protocol was overdue. Other interesting facts about IPv6 are the better unicast and broadcasting methods, usage of hexadecimal numbers in the IP address format, and the elimination of decimal dots with the introduction of colons (":") as delimiters in the address.

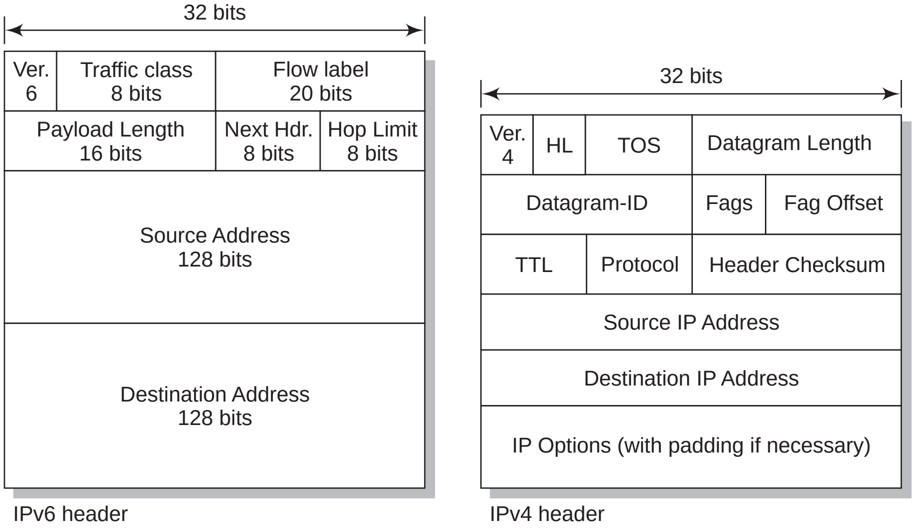


Figure 8.1: IPv6 and IPv4 header formats

8.1 IPV6 PACKET HEADER

The IPv6 packet contains the header and the payload. The header part contains the information required for the packet to reach its destination. The payload is the actual data. The IPv6 header is surprisingly simpler than the IPv4 header.

To understand the IPv6 header, let us see a pictorial comparison of the IPv4 and IPv6 headers.

8.1.1 Packet Header Structure

The header is of length ***40 bytes*** and has eight fields. Some of these fields could be seen in the IPv4 header as well. The various fields are:

1. **Version:** Informs the routers which Internet Protocol version is in use. It would be 6 for IPv6.
2. **Traffic Class (8 bits):** This field is similar to the ***Type of service*** field in IPv4. It gives the class of service (CoS) priority of the packet. Using this field, devices can distinguish between latency sensitive traffic (voice, video etc) and low-priority data (email, web traffic etc).
3. **Flow Label (20 bits):** The host uses this field so that special handling is obtained from the routers that are IPv6-compliant. It indicates which packets are a part of a special flow or the ones that need a special class of service (CoS). This management of traffic flow between stations helps in giving a quality of service.
4. **Payload Length (16 bits):** Describes the length, in octets, of the IPv6 payload (data) part of the IPv6 packet. The 16-bit field length (2^{16}) lets version 6 support payloads in excess of 64,000 octets.
5. **Next Header (8 bits):** This field was initially the protocol field in IPv4. It alerts routers about the next extension header that should be examined.
6. **Hop limit (8 bits):** Specifies the number of routing hops a packet is allowed to take before being discarded. The field is 8 bits in length, hence the maximum hops a packet can take is 255. No path would be that long in today's networks.
7. **Source Address (128 bits):** Gives the 128 bit address of the source node.
8. **Destination Address (128 bits):** Specifies the 128-bit address of the final destination node.

8.1.2 Extension Headers

There is the option of placing extension headers in between the IPv6 header and the upper-layer header of the packet for optional Internet layer information like source routing, encryption, and authentication. The extension headers, if more than one, can be held together by using the **Next Header field** in the IPv6 header. It informs the router which extension header should be next in line.

If there are no extension headers, the ***Next header*** field would indicate the upper layer header like the TCP header, UDP header, etc.

8.1.3 Addressing Description

The IPv6 address is 128 bits long, or 16 bytes (octets). The address is specified in a ***hexadecimal colon notation.*** Hexadecimal numbers are base 16. The following figure describes the how the IPv6 address is represented. The 128 bits are divided into eight parts, with each part being 2 bytes (16 bits) in length. Converting 2 bytes to the hexadecimal format would become four hexadecimal digits. This means that the 128 bit address in a hexadecimal format would have 32 hexadecimal digits. Notice the colon after every four hexadecimal digits.

Therefore, each part *x* would be a 16-bit binary number and there are eight parts in total. This would mean that the total length would be equal to 16 * 8 = 128 bits.

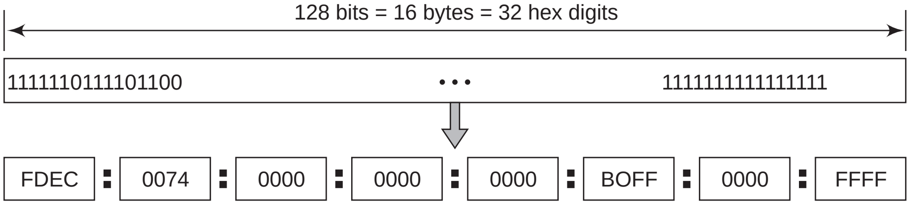


Figure 8.2(a): IPv6 address in hexadecimal format

8.1.4 Abbreviated IPv6 Address

Even after conversion to the hexadecimal format, the IPv6 address is really long. But there are many cases where the digits are zeroes, and then the address can be abbreviated or shortened. The ***leading zeroes*** can be left out thereby shortening the IP address. Remember that no trailing zero can be dropped.

Numbers like 0005 can be reduced to just 5, and 0000 as 0. But numbers like 5500 cannot be shortened. If there are consecutive parts consisting of zeroes only, then the zeroes can be removed and replaced by just a semicolon. But this is a one-time only possibility in an address. This means that even if there are two parts of only zeroes in an address, only one of them can be shortened to a semi-colon. The following diagram gives an example of an abbreviated IPv6 address.

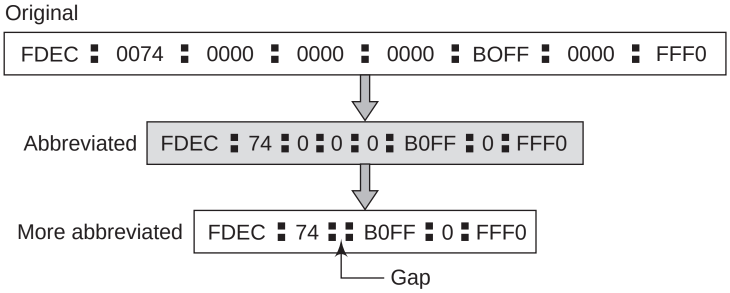

Original
FDEC : 0074 : 0000 : 0000 : 0000 : BOFF : 0000 : FFF0
Abbreviated
FDEC : 74 : 0 : 0 : 0 : B0FF : 0 : FFF0
More abbreviated
FDEC : 74 : : B0FF : 0 : FFF0
Gap

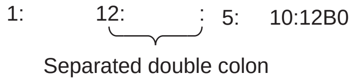
Figure 8.2(b): Abbreviated IPv6 address

The question arises as to how can one get the original address from the abbreviated address? This can be understood by an example:

Abbreviated IPv6 address: **1:12::5:10:12B0**

Now, align the left part of the double colon to the left hand side of an original hexadecimal format of the IPv6 address and the right side of the double colon to the right side of the original format to understand how many parts need to be replaced by zeroes.

Example of an IPv6 address: xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx



Original address would be: 0001:0012:0000:0000:0000:0005:0010:12B0

8.2 BROADCASTING METHODS

Since the IPv6 address is 128 bits long, there are $\mathbf{2^{128}}$ addresses available to be assigned to hosts on the Internet. The address space thus becomes much larger than what was available for the IPv4 addresses.

The IPv6 address has been categorized into several types with some of its leftmost bits being the defining factor. These leftmost bits, referred to as the ***type prefix*** is not fixed in length but ensures that no code is similar to the beginning part of any other code. This eliminates any uncertainty that could occur as one can just look at an address and determine its type prefix. The following table shows the type prefixes with the type of address.

Table 8.1: Type prefixes for address types

Type Prefix	Type
0000 0000	Reserved
0000 0001	Unassigned
0000 001	ISO network addresses
0000 010	IPX (Novell) network addresses
0000 011	Unassigned
0000 1	Unassigned
0001	Reserved
001	Reserved
010	**Provider-based unicast addresses**
011	Unassigned
100	Geographic based unicast addresses
101	Unassigned
110	Unassigned
1110	Unassigned
1111 0	Unassigned
1111 10	Unassigned
1111 110	Unassigned
1111 1110 0	Unassigned
1111 1110 10	Link Local addresses
1111 1110 11	Site local addresses
1111 1111	Multicast addresses

IPv6 includes some new broadcasting methods:

1 Unicast
2 Multicast
3 Anycast

In the case of IPv4 addresses, it was never understandable whether an address was assigned to a node or a network interface. When it came to IPv6, designers made sure that an address was defined or configured on each network interface.

8.2.1 Unicast Addresses

Any packet being sent to a unicast address will be received by a specific computer. As the name suggests, a unicast address is the address of one computer. There can be two types of unicast addresses: provider-based and geographically based. Since the geographically based unicast addresses are reserved for future use, we will discuss provider-based unicast addresses.

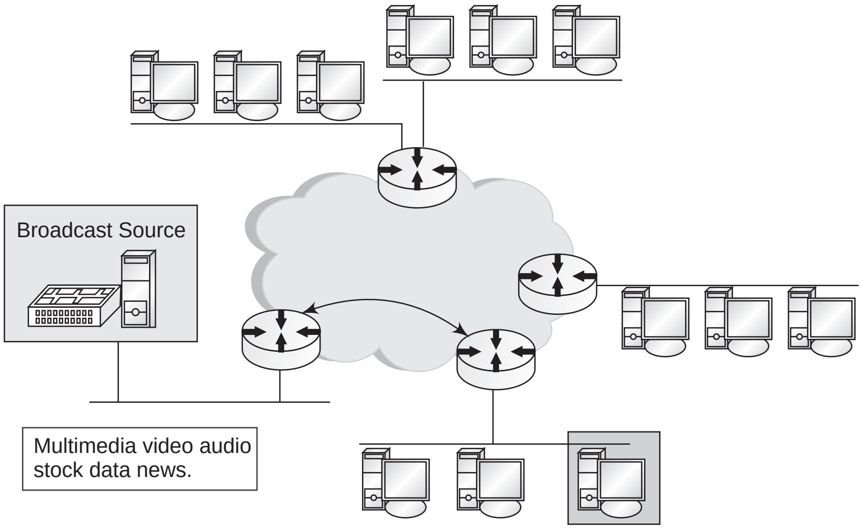


Figure 8.3: Broadcasting over unicast addresses

The following figure shows the prefixes for a provider-based unicast address.

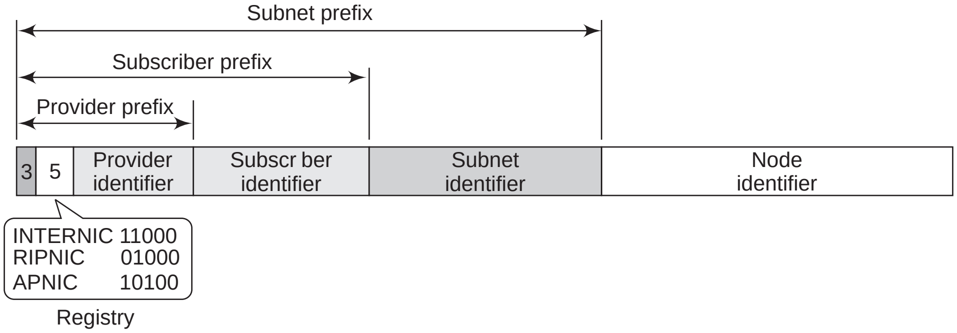


Figure 8.4: Prefixes for a provider-based unicast address

1. **Type identifier (3 bits):** The first three bits specify that the address is provider-based. Referring to the previous table which showed the type prefixes, these three bits would be 010.
2. **Registry Identifier (5 bits):** Identifies the registry center that has registered the address.

Registry Center	Code
INTERNIC (North American Center)	**11000**
RIPNIC (Europe)	**01000**
APNIC (Asia-Pacific)	**10100**

3. **Provider Identifier (Variable length, recommended-16 bits):** Specifies the Internet access provider.

4. **Subscriber identifier (recommended-24 bits):** Identifier given to the organization that subscribes to the Internet via a provider.
5. **Subnet identifier (recommended-32 bits):** Specifies the subnetwork in the subscriber's territory.
6. **Node Identifier (recommended- 48 bits):** Identifies the node in the subnet. The node identifier may Equate the length of the node's physical address in the future.

8.2.2 Multicast Addresses

A multicast address is used to send packets to a group of computers. This means that multiple hosts may be registered to a multicast group, and this group has a multicast address assigned to it. Thus, by specifying just the multicast address, the sending host can send a packet to all hosts in that particular multicast group.

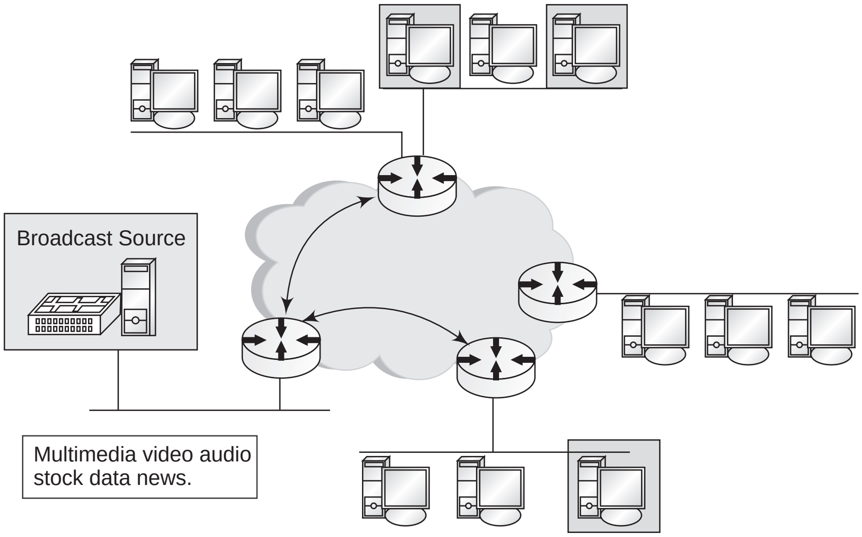


Figure 8.5: Broadcasting over multicast addresses

The multicast address format is shown in the following diagram. As specified in the type prefix table, multicast addresses will have the type prefix as 1111 1111.

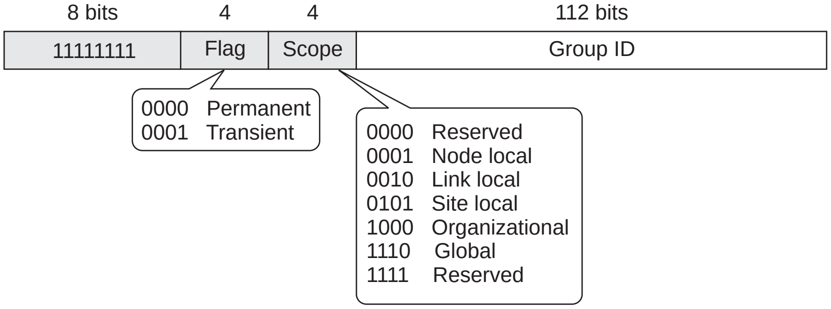


Figure 8.6: Type prefix for multicast addresses

Flag (4 bits): Indicates if the group address is permanent (0000) or transient (0001). Transient group addresses are temporary addresses whereas permanent group addresses are accessible all the time. They are specified by the Internet authorities.

Scope (4 bits): Defines the scope of the group address.

8.2.3 Anycast Addresses

The anycast address defines a set of hosts, just like the multicast address, but when packets are sent, it is delivered to the nearest interface defined by that address. So, in contrast to transmitting the packet to all nodes in a group as in the case of multicast, anycast would only send the packet to the nearest node in the anycast group.

Internet Protocol (IP) multicast is a routing technique that allows IP traffic to be sent from one source or multiple sources and delivered to multiple destinations. Instead of sending individual packets to each destination, a single packet is sent to a multicast group, which is identified by a single IP destination group address. IP multicast routing arose because unicast and broadcast techniques do not handle the requirements of new applications efficiently. Multicast addressing, for example, supports the transmission of a single IP datagram to multiple hosts. This chapter focuses on the leading multicast routing options. Figure illustrates the general nature of a multicast environment.

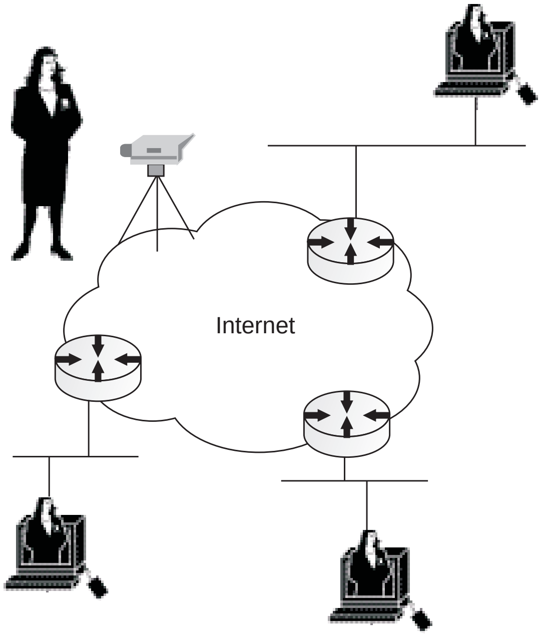


Figure 8.7: IP multicast provides a means to deliver high-bandwidth traffic to multiple destinations

8.3 MULTICAST ROUTING PROTOCOLS

Multicast link state routing uses the source-based tree approach.

Multicast routing protocols enable a collection of multicast routers to build (join) distribution trees when a host on a directly attached subnet, typically a LAN, wants to receive traffic from a certain multicast group.

There are basically four multicast routing protocols that can be classified as follows:

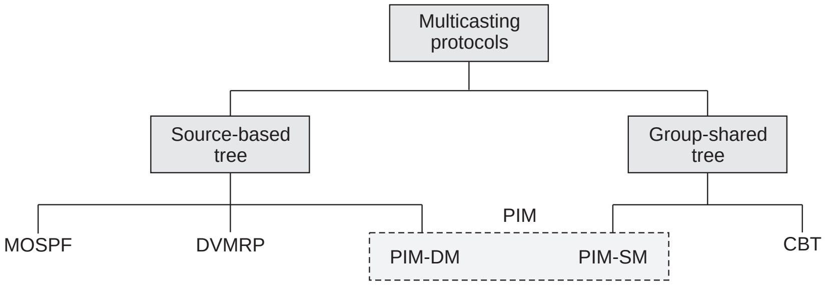


Figure 8.8: Multicasting routing protocols

Let us first see the *source-based* tree protocols:

1. DVMRP—The **Distance Vector Multicast Routing Protocol** (**DVMRP**) is used to share information between routers to transport IP Multicast packets among networks. It is based on the RIP protocol to forward packets: the router generates a routing table with the multicast group that it has knowledge with its corresponding distance (number of devices -routers- in the middle to reach it).

   When a Multicast packet is received by a router, it is forwarded by router's interfaces specified in the routing table. The DVMRP protocol uses IGMP messages to exchange information with other routers.

   DVMRP uses IGMP to exchange routing datagrams. IGMP Header is a collection of 16 bits, which can be categorized, as follows:

   First 4 bits for **version**, Next 4 bits for **Type** and the remaining 8 bits for *subtype* which may be any of the following:

   1 = Response; 2 = Request; 3 = Non-membership report and
   4 = Non-membership cancellation;

   DVMRP is a dense-mode-only protocol, and uses the flood-and-prune or implicit join method to deliver traffic everywhere and then determine where the uninterested receivers are.

2. MOSPF— **Multicast Open Shortest Path First** (**MOSPF**) protocol is an extension to the Open Shortest Path First (OSPF) protocol to support multicast routing, allowing routers to share information about group memberships.However, MOSPF has an explicit join message, so routers do not have to flood their entire domain with multicast traffic from every source.

3. PIM— **Protocol-Independent Multicast** (PIM) is a family of multicast routing protocols for Internet Protocol (IP) networks that provide one-to-many and many-to-many distribution of data over a LAN, WAN or the Internet.

   It is termed protocol-independent because PIM does not include its own topology discovery mechanism, but instead uses routing information supplied by other traditional routing protocols such as the Border Gateway Protocol (BGP).

   However, PIM is categorized among both source-based and group-shared tree structure for multicasting routing protocols. There are four variants of PIM:

   - **PIM Sparse Mode** (PIM-SM) explicitly builds unidirectional shared trees rooted at a rendezvous point (RP) per group, and optionally creates shortest-path trees per source. PIM-SM generally scales fairly well for wide-area usage.
   - **PIM Dense Mode** (PIM-DM) uses dense multicast routing. It implicitly builds shortest-path trees by flooding multicast traffic domain wide, and then pruning back branches of the tree where no receivers are present.

- PIM-DM is straightforward to implement but generally has poor scaling properties. The first multicast routing protocol, DVMRP used dense-mode multicast routing
- **Bidirectional PIM** explicitly builds shared bi-directional trees. It never builds a shortest path tree, so may have longer end-to-end delays than PIM-SM, but scales well because it needs no source-specific state.
- **PIM source-specific multicast (PIM-SSM)** builds trees that are rooted in just one source, offering a more secure and scalable model for a limited amount of applications (mostly broadcasting of content). In SSM, an IP datagram is transmitted by a source S to an SSM destination address G, and receivers can receive this datagram by subscribing to channel (S,G).

Table 8.2: Characteristics of multicast routing protocols

Protocol	**Unicast Protocol Requirements**	**Flooding Algorithm**
PIM-dense mode	Any	Reverse path flooding (RPF)
PIM-sparse mode	Any	RPF
DVMRP	Internal, RIP-like routing protocol	RPF
MOSPF	Open Shortest Path First (OSPF)	Shortest-path first (SPF)

It is important to realize that retransmissions due to a high bit-error-rate on a link or overloaded router can make multicast as inefficient as repeated unicast. Therefore, there is a trade-off in many multicast applications regarding the session support when using TCP (but TCP always resends missing segments) or the simple drop-and-continue strategy of the UDP datagram service (but reordering can become an issue). Modern multicast uses UDP almost exclusively.

SUMMARY

- IPv6 expanded the addressing from 32-bit (4octets) addresses to 128-bit (16 octets) addresses.
- The IPv6 address is 128 bits long, or 16 bytes (octets) and is specified in a ***hexadecimal colon notation.***
- IPv6 includes some new broadcasting methods: **Unicast, Multicast, and Anycast**

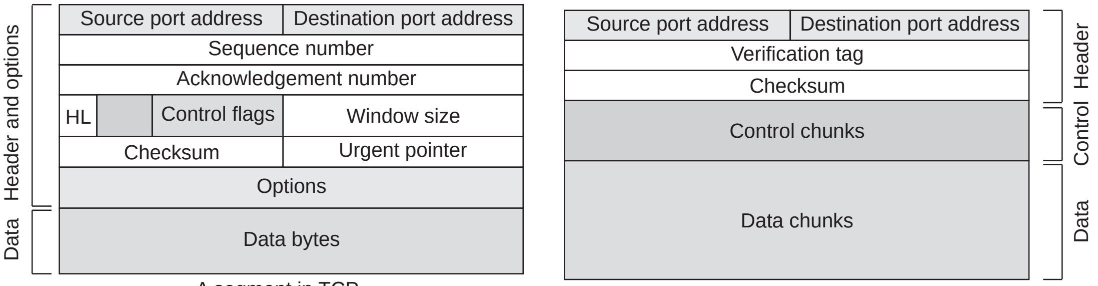
9

SCTP

Introduction

The Stream Control Transmission Protocol (SCTP) is a Transport Layer protocol, serving in a similar role to the popular protocols Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). It provides some of the same service features of both: it is message-oriented like UDP and ensures reliable, in-sequence transport of messages with congestion control like TCP.

The protocol was defined by the IETF Signaling Transport (SIGTRAN) working group in 2000, and is maintained by the IETF Transport Area (TSVWG) working group. RFC 4960 defines the protocol.



A segment in TCP

A packet in SCTP

Figure 9.1: Comparing SCTP packet and TCP segment formats

9.1 SCTP

Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced. These new applications need a more sophisticated service than TCP can provide.

TCP has segments; SCTP has packets

It distinguishes different streams of messages within one SCTP association. This enables a delivery scheme where only the sequence of messages needs to be maintained per stream (partial in-sequence delivery) which reduces unnecessary head-of-line blocking between independent streams of messages. Furthermore, SCTP provides a mechanism for bypassing the sequenced delivery service, so that messages are delivered to the user of SCTP as soon as they are completely received (order-of-arrival delivery).

Flow control and congestion control in SCTP have been designed in a way which assures that SCTP traffic behaves in the network in the same way as TCP traffic does. This enables a seamless introduction of SCTP services into existing IP networks.

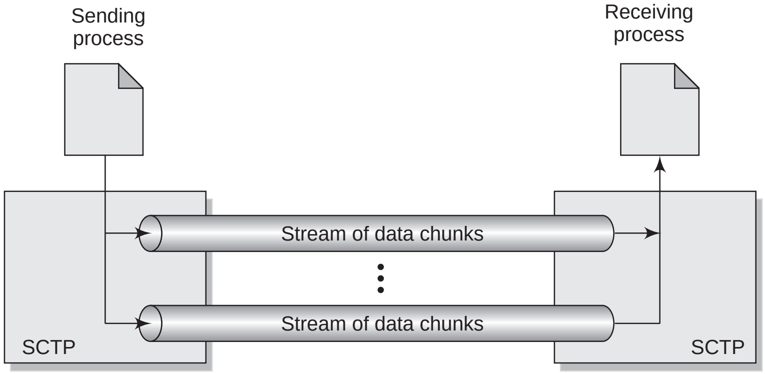


Figure 9.2: Data transfer in SCTP

SCTP operates on two levels:

- Within an association the reliable transfer of datagrams is assured by using a checksum, a sequence number and a selective retransmission mechanism. Without taking the initial sequence into account, every correctly received data chunk is delivered to a second, independent level.
- The second level realises a flexible delivery mechanism which is based on the notion of several independent streams of datagrams within an association.

Detection of loss and duplication of data chunks is enabled by numbering all data chunks in the sender with the so-called Transport Sequence Number (TSN). The acknowledgements sent from the receiver to the sender are based on these sequence numbers. Retransmissions are timer-controlled. The timer duration is derived from continuous measurements of the round trip delay. Whenever such a retransmission timer expires, (and congestion control allows transmissions) all non-acknowledged data chunks are retransmitted and the timer is started again doubling its initial duration (like in TCP). When the receiver detects one or more gaps in the sequence of data chunks, each received SCTP packet is acknowledged by sending a Selective Acknowledgement (SACK) which reports all gaps. The SACK is contained in a specific control chunk. Whenever the sender receives four consecutive SACKs on the same data chunk this data chunk is immediately retransmitted (fast retransmit).

9.2 FEATURES

Features of SCTP include:

- Multihoming support in which one or both endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between redundant network paths.

- Delivery of chunks within independent streams eliminates unnecessary head-of-line blocking, as opposed to TCP byte-stream delivery.
- Path selection and monitoring select a primary data transmission path and test the connectivity of the transmission path.
- Validation and acknowledgement mechanisms protect against flooding attacks and provide notification of duplicated or missing data chunks.
- Improved error detection suitable for Ethernet jumbo frames.

9.3 SCTP PACKETS

SCTP Packets consist of a simple common header and one or more chunks. The Header consists of a *Source Port* number, a *Destination Port* number, a *Checksum* and a *Verification* tag. Each of these except the Verification tag is found in both UDP and TCP. The verification attack prevents attacks by making it difficult for packets to be injected into an active stream. It is simply a number that is used to further confirm that the packet was sent by the other end-point in the Association.

9.3.1 SCTP Header Format

Source port address 16 bits	Destination port address 16 bits
Verification tag 32 bits	
Checksum 32 bits	

Figure 9.3: SCTP header format

Source port. 16 bits.

The SCTP sender's port number. It can be used by the receiver in combination with the source IP address, the SCTP destination port and possibly the destination IP address to identify the association to which this packet belongs.

Destination port. 16 bits.

The SCTP port number to which this packet is destined. The receiving host will use this port number to de-multiplex the SCTP packet to the correct receiving endpoint/application.

Verification tag. 32 bits.

The receiver of this packet uses the Verification Tag to validate the sender of this SCTP packet. On transmit, the value of this Verification Tag MUST be set to the value of the Initiate Tag received from the peer endpoint during the association initialization, with the following exceptions:

- A packet containing an INIT chunk MUST have a zero Verification Tag.
- A packet containing a SHUTDOWN-COMPLETE chunk with the T-bit set MUST have the Verification Tag copied from the packet with the SHUTDOWN-ACK chunk.
- A packet containing an ABORT chunk may have the verification tag copied from the packet which caused the ABORT to be sent.

An INIT chunk MUST be the only chunk in the SCTP packet carrying it.

Checksum: 32 bits.

Contains the checksum of this SCTP packet. SCTP uses the Adler-32 algorithm for calculating the checksum.

9.3.2 Packet Structure

Each packet consists of one or more *Chunks*. In SCTP, control information and data information are carried in separate chunks. Each function in SCTP has a different chunk.

Chunk type. It is a set of 8 bits from 0 to 255 that Identifies the type of information contained in the Chunk data. The value of 255 is reserved for future use as an extension field.

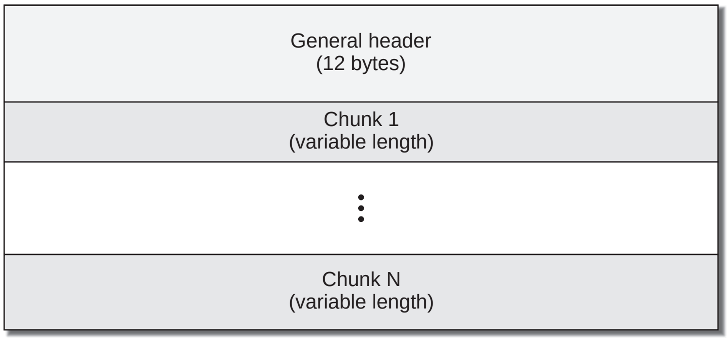


Figure 9.4: Chunks in an SCTP packet

Chunk types are encoded such that the highest-order two bits specify the action that must be taken if the processing endpoint does not recognize the Chunk type.

- 00 – Stop processing this SCTP packet and discard it, do not process any further chunks within it.
- 01 – Stop processing this SCTP packet and discard it, do not process any further chunks within it, and report the unrecognized parameter in an 'Unrecognized Parameter Type' (in either an ERROR or in the INIT ACK).
- 10 – Skip this chunk and continue processing.
- 11 – Skip this chunk and continue processing, but report in an ERROR Chunk using the 'Unrecognized Chunk Type' cause of error.

The total length of a chunk MUST be a multiple of 4 bytes. If the length of the chunk is not a multiple of 4 bytes, the sender MUST pad the chunk with all zero bytes and this padding is not included in the chunk length field. The sender should never pad with more than 3 bytes. The receiver MUST ignore the padding bytes.

The original standard defined 14 Chunk types.

Table 9.1: Chunk types

Type	Chunk	Description
0	DATA	User data
1	INIT	Sets up an association
2	INIT ACK	Acknowledges **INIT** chunk
3	SACK	Selective acknowledgement
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Aborts an association

7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIE ACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARD TSN	For adjusting cumulative TSN

Data chunks are identified by three items: TSN, SI, and SSN.

TSN is a cumulative number identifying the association; SI defines the stream; SSN defines the chunk in a stream.

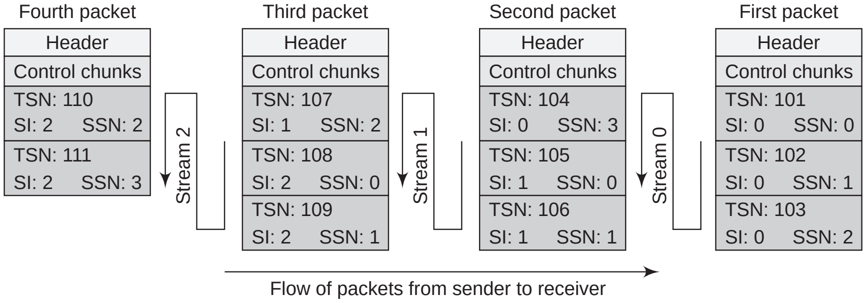


Figure 9.5: Use of chunks during transfer of packets

9.4 ASSOCIATION

A connection in SCTP is called an association No other chunk is allowed in a packet carrying an INIT or INIT ACK chunk. A COOKIE ECHO or a COOKIE ACK chunk can carry data chunks.

Like TCP uses 3-way Handshake, SCTP uses a 4-way Handshake protocol for establishment of a connection, also called Association. In SCTP, only DATA chunks consume TSNs; DATA chunks are the only chunks that are acknowledged.

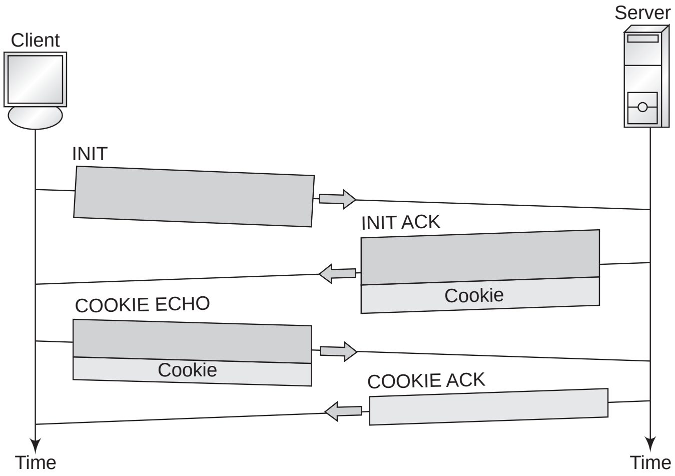


Figure 9.6(a): 4-way handshake for association in SCTP

Following is an example of data transfer using SCTP:

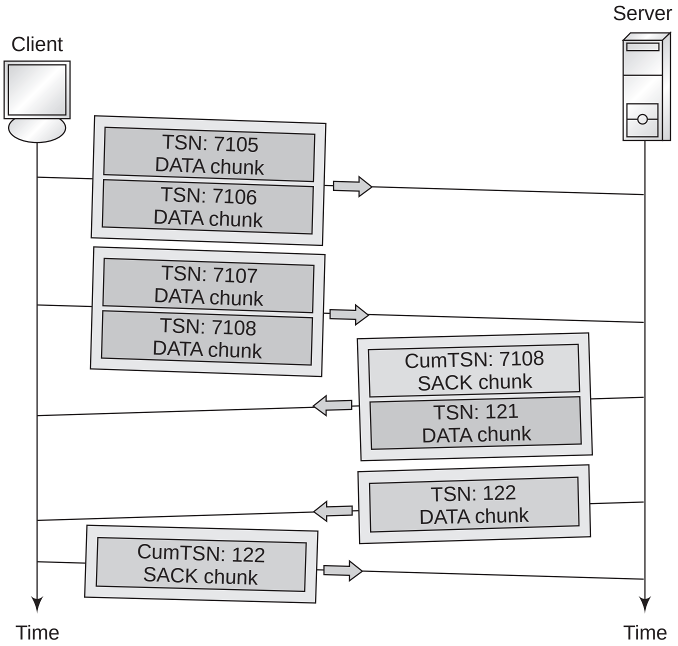


Figure 9.6(b): Data transfer using SCTP

The acknowledgement in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order.

Terminating an Association

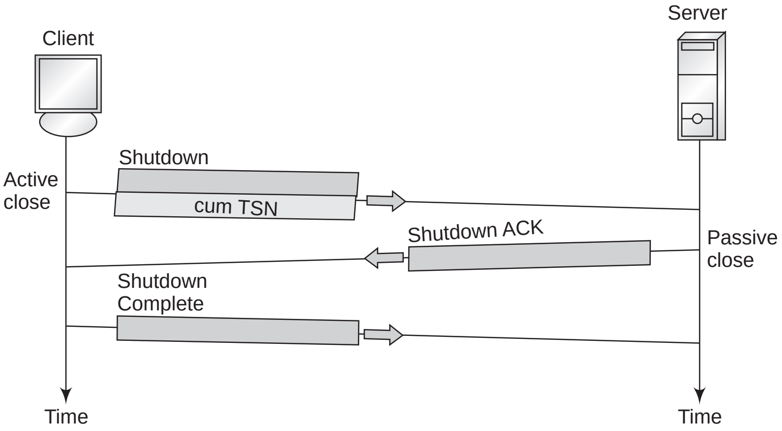


Figure 9.6(c): Termination of an association

9.5 FLOW CONTROL

SCTP uses an end-to-end window based flow and congestion control mechanism similar to the one that is well known from TCP .The receiver of data may control the rate at which the sender is sending by specifying an octet-based window size (the so-called Receiver Window), and returning this value along with all SACK chunks.

The sender itself keeps a variable known as Congestion Window (short: CWND) that controls the maximum number of outstanding bytes (i.e. bytes that may be sent before they are acknowledged). Each received data chunk must be acknowledged, and the receiver may wait a certain time (usually 200 ms) before that is done. If there are larger number of SCTP packets with data received within this period of, every second SCTP packet containing data is to be acknowledged at once by sending a SACK chunk back to the sender.

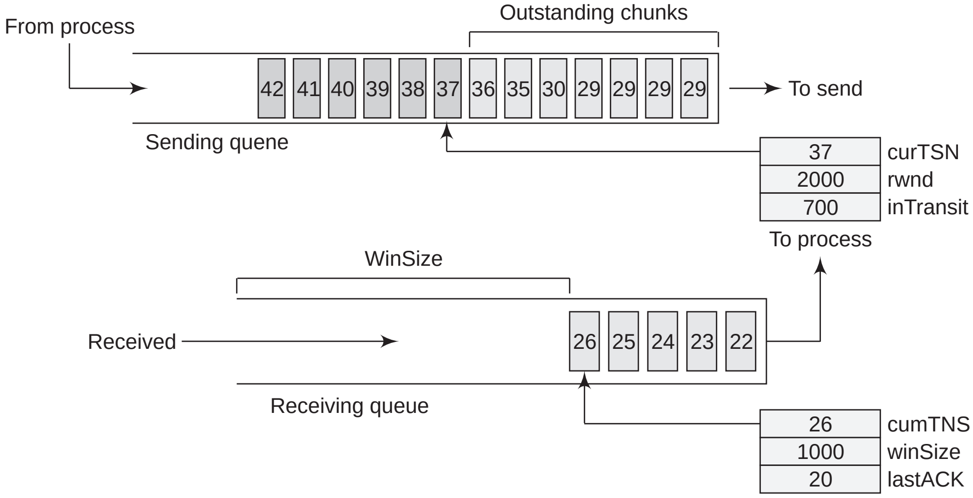


Figure 9.7(a): Congestion window in SCTP

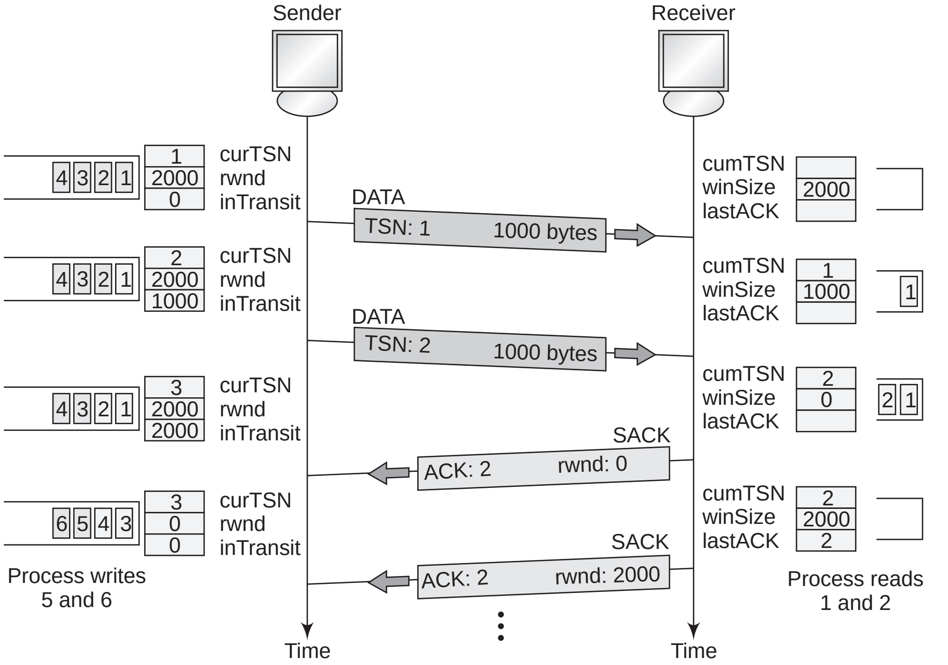


Figure 9.7(b): Example showing use of congestion window for flow control in SCTP

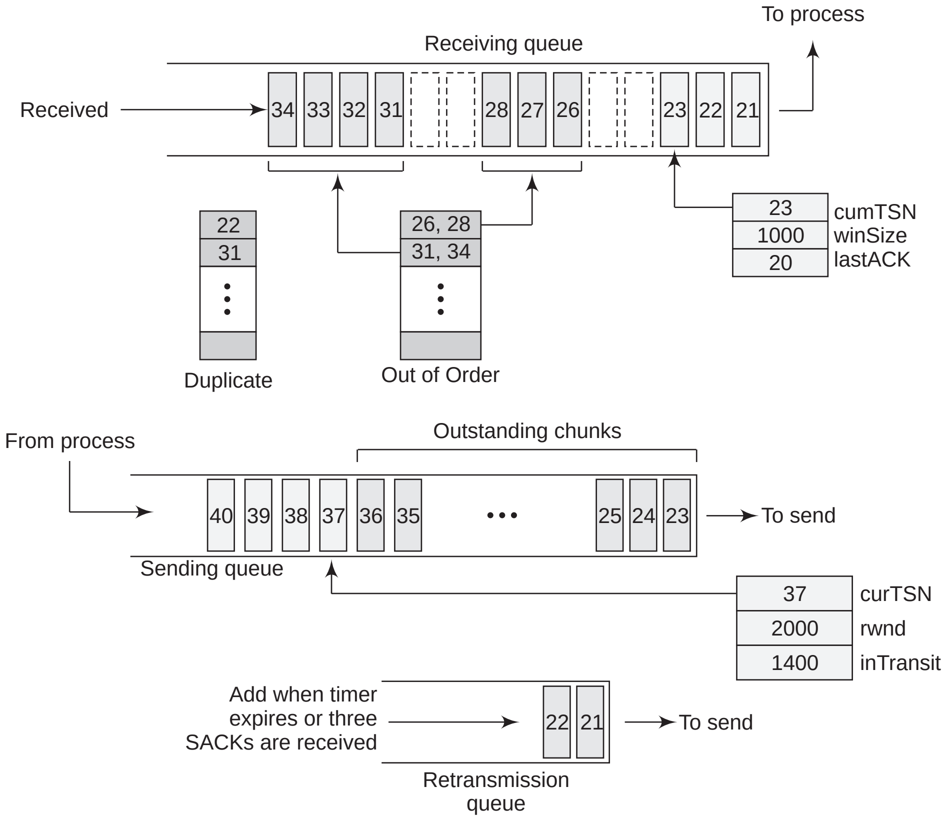


Figure 9.8: Selective acknowledgement

9.5.1 Selective Acknowledgement

The acknowledgements carry all TSN numbers that have been received by one side with them. That is, there is a so called **Cumulative TSN Ack** value, that indicates all the data that has successfully been reassembled at the receivers side, and has either already been delivered to the receiving Upper Layer Process, or may readily be delivered upon request.

Moreover, there are so-called **Gap Blocks** that indicate that segments of data chunks have arrived, with some data chunks missing in between. Should some data chunks have been lost in the course of transmission, they will either be retransmitted after the transmission timer has expired, or after four SACK chunks have reported gaps with the same data chunk missing. In the latter case, the missing data is retransmitted via the **Fast Retransmit** mechanism.

In case a retransmission occurs which signals packet loss, the implementation must appropriately update congestion and flow control parameters.

9.6 CONGESTION CONTROL

The congestion control behaviour of an SCTP implementation according to RFC2960 may have an impact where timely delivery of messages is required (i.e. transport of signalling data). However, this ensures the proper behaviour of SCTP when it is introduced on a large scale into existing packet switched networks such as the Internet. The congestion control mechanismns for SCTP have been derived from RFC 2581 - TCP Congestion Control), and been adapted for multihoming. For each destination address (i.e. each possible path) a discrete set of flow and congestion control parameters is kept, such that from the point of view of the network, an SCTP association with a number of paths may behave similarly as the same number of TCP connections.

9.6.1 Slow Start and Congestion Avoidance

As in TCP, SCTP has two modes, Slow Start and Congestion Avoidance. The mode is determined by a set of congestion control variables, and as already mentioned, these are path specific. So, while the transmission to the primary path may be in the Congestion Avoidance mode, the implementation may still use Slow Start for the backup path(s).

For successfully delivered and acknowledged data the congestion window variable (CWND) is steadily increased, and once it exceeds a certain boundary (called Slow Start Threshold, SSTRESH), the mode changes from Slow Start to Congestion Avoidance. Generally, in Slow Start, the CWND is increased faster (roughly one MTU per received SACK chunk), and in Congestion Avoidance mode, it is only increased by roughly one MTU per Round Trip Time (RTT).

Events that trigger retransmission (timeouts or fast retransmission) cause the SSTHRESH to be cut down drastically, and reset the CWND (where a timeout causes a new Slow Start with CWND=MTU, and a Fast Retransmit sets CWND=SSTHRESH.

SUMMARY

- SCTP is a message-oriented, reliable protocol that combines the best features of UDP and TCP.
- An association in SCTP can involve multiple streams
- SCTP association allows multiple IP addresses for each end.
- In SCTP, a data chunk is numbered using a TSN.
- To distinguish between different streams, SCTP uses an SI
- To distinguish between different data chunks belonging to the same stream, SCTP uses SSNs.

10

CONGESTION CONTROL AND QOS

Introduction

As Internet can be considered as a *Queue of packets,* where transmitting nodes are constantly adding packets and some of them (receiving nodes) are removing packets from the queue. So, consider a situation where too many packets are present in this queue (or internet or a part of internet), such that constantly transmitting nodes are pouring packets at a higher rate than receiving nodes are removing them. This degrades the performance, and such a situation is termed as *Congestion*. The main reason of congestion is that more number of packets are introduced into the network than it can handle.

So, the objective of congestion control can be summarized as **to maintain the number of packets in the network below the level at which performance falls** off dramatically. The nature of a Packet switching network can be summarized in following points:

- A network of queues
- At each node, there is a queue of packets for each outgoing channel
- If packet arrival rate exceeds the packet transmission rate, the queue size grows without bound
- When the line for which packets are queuing becomes more than 80% utilized, the queue length grows alarmingly

When the number of packets dumped into the network is within the carrying capacity, they all are delivered, expect a few that have too be rejected due to transmission errors). And then the number delivered is proportional to the number of packets sent.

However, as traffic increases too far, the routers are no longer able to cope, and they begin to lose packets. This tends to make matter worse. At very high traffic, performance collapse completely, and almost no packet is delivered.

In the following sections, the causes of congestion, the effects of congestion and various congestion control techniques are discussed in detail.

10.1 CONGESTION

Congestion can occur due to several reasons. For example, if all of a sudden a stream of packets arrives on several input lines and needs to be out on the same output line, then a long queue will be build up for that output. If there is *insufficient memory* to hold these packets, then some packets will be lost (dropped). Adding more memory also may not help in certain situations.

If the router has an infinite amount of memory even then instead of congestion being reduced, it gets worse; because by the time packets gets at the head of the queue, to be dispatched out to the output line, they have already timed-out (repeatedly), and duplicates may also be present.

All the packets will be forwarded to next router up to the destination, all the way only increasing the load to the network more and more. Finally when it arrives at the destination, the packet will be discarded, due to time out, so instead of being dropped at any intermediate router (in case memory is restricted) such a packet goes all the way up to the destination, increasing the network load throughout and then finally gets dropped there.

Slow processors also cause Congestion. If the router CPU is slow at performing the task required for them (Queuing buffers, updating tables, reporting any exceptions etc.), a queue can build up even if there is excess of line capacity. Similarly, *Low-Bandwidth* lines can also cause congestion. Upgrading lines but not changing slow processors, or vice-versa, often helps a little; these can just shift the bottleneck to some other point. The real problem is the mismatch between different parts of the system.

Congestion tends to feed upon itself to get even worse. Routers respond to overloading by dropping packets. When these packets contain TCP segments, the segments don't reach their destination, and they are therefore left unacknowledged, which eventually leads to timeout and retransmission.

So, the major cause of congestion is often the *bursty* nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion because other causes just act as an enzyme which boosts up the congestion when the traffic is bursty (i.e., other causes just add on to make the problem more serious, main cause is the bursty traffic).

This means that when a device sends a packet and does not receive an acknowledgment from the receiver, in most the cases it can be assumed that the packets have been dropped by intermediate devices due to congestion. By detecting the rate at which segments are sent and not acknowledged, the source or an intermediate router can infer the level of congestion on the network.

In the following section we shall discuss the ill effects of congestion.

10.1.1 Effects of Congestion

Congestion affects two vital parameters of the network performance, namely *throughput* and *delay*. In simple terms, the throughput can be defined as the percentage utilization of the network capacity. The following figure shows how throughput is affected as offered load increases. Initially throughput increases linearly with offered load, because utilization of the network increases. However, as the offered load increases beyond certain limit, say 60% of the capacity of the network, the throughput drops. If the offered load increases further, a point is reached when not a single packet is delivered to any destination, which is commonly known as *deadlock* situation.

There are three curves in the figure, where the ideal one corresponds to the situation when all the packets introduced are delivered to their destination up to the maximum capacity of the network.

The second one corresponds to the situation when there is no congestion control. The third one is the case when some congestion control technique is used. This prevents the throughput collapse, but provides lesser throughput than the ideal condition due to overhead of the congestion control technique.

The delay also increases with offered load and, no matter what technique is used for congestion control, the delay grows without bound as the load approaches the capacity of the system.

It may be noted that initially there is longer delay when congestion control policy is applied. However, the network without any congestion control will saturate at a lower offered load.

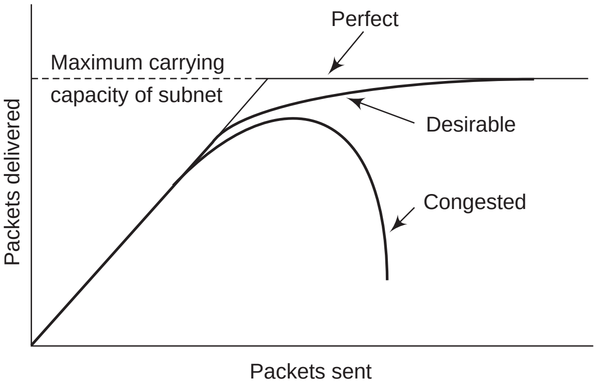


Figure 10.1: Throughput and congestion in a network

10.1.2 General Principles of Congestion Control

Congestion control, as the name suggests, comprises of several methods, or policies, that would govern the transfer of data packets over a network.

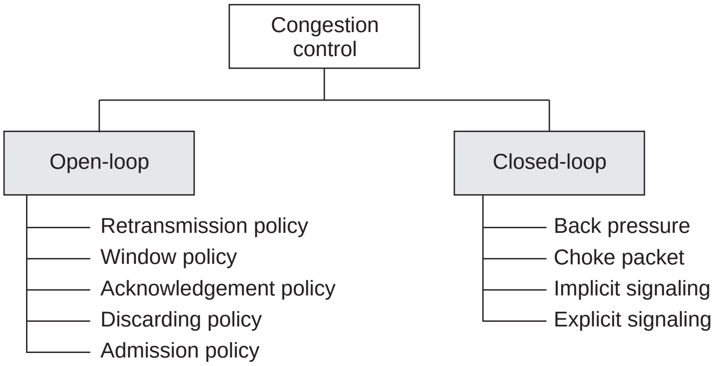


Figure 10.2: Classification of congestion control policies

From a control theory point of view, all solutions to problems in complex systems, such as computer networks, can be divided into two groups:

- **Open loop** solutions solve the problem by good design, in essence, to make sure the problem does not occur in the first place.

Tools include deciding when to accept new traffic, when to discard packets and which ones, and how to schedule packets at various points in the network.

A common fact: they make decisions without regard to the current state of the network.

- **Closed loop** solutions are based on the concept of a feedback loop, which consists of the following three parts:
  1. Monitor the system to detect when and where congestion occurs.
  2. Pass this information to places where actions can be taken.
  3. Adjust system operation to correct the problem.

The closed loop algorithms can also be divided into two categories, namely *explicit feedback* and *implicit feedback* algorithms. In the explicit approach, special packets are sent back to the sources to curtail down the congestion. While in implicit approach, the source itself acts pro-actively and tries to deduce the existence of congestion by making local observations.

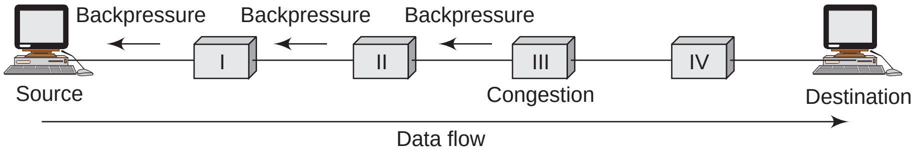
Now, let us see each of these policies, in detail:

Open-Loop Policies

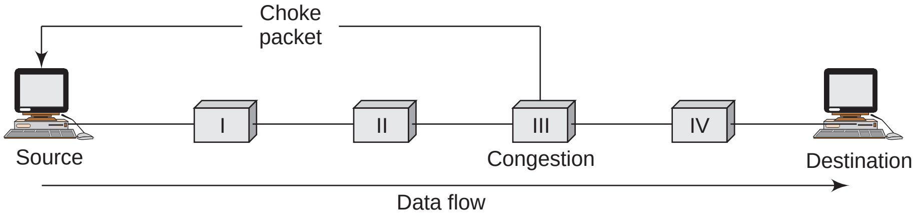
- Retransmission policy: Network Congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Network protocols which use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion even after the initial load has been reduced to a level which would not normally have induced network congestion. The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.
- Window policy: Analogous to the mechanism used by Sliding Window Protocol, we need to define an upper limit to the *windowing* size for that network. This is established through the window policy for the system.
- Acknowledgement policy: If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.
- Discard policy: In audio transmission, if the policy is to discard less sensitive packets when congestion is likely, the quality of sound is still preserved and congestion is prevented.
- Admission policy: It is a function applied on a per-packet basis to make sure that a flow conforms to the Type of Service that was used to make the reservation.

Closed-Loop Policies

- Back pressure: This method involves informing the previous upstream router to reduce the rate of outgoing packets.
- Choke packet: It is a packet sent by a router to the source to inform it of congestion. This is similar to ICMP's source quench packet.



(a) Back pressure



(b) Choke packet

Figure 10.3: Closed-loop congestion control policies

- Implicit signaling: Detecting an implicit signal warning of congestion and slowing down its sending rate. For example, receiving delayed ACK
- Explicit signaling: Router experiencing congestion can send an explicit signal by setting a bit in a packet to the sender or the receiver.

10.2 CONGESTION CONTROL IN TCP

The TCP uses a network congestion avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) scheme, with other schemes such as slow-start in order to achieve **congestion avoidance**. Also related to TCP congestion control is the slow start mechanism.

Slow-start is part of the congestion control strategy used by TCP, the data transmission protocol used by many Internet applications. Slow-start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, to avoid causing network congestion.

10.2.1 Congestion Control Algorithms

Slow-start is one of the algorithms that TCP uses to control congestion inside the network. It is also known as the exponential growth phase.

During the exponential growth phase, Slow-start works by increasing the TCP congestion window each time the acknowledgement is received. It increases the window size by the number of segments acknowledged. This happens until either an acknowledgement is not received for some segment or a predetermined threshold value is reached. If a loss event occurs, TCP assumes this it is due to network congestion and takes steps to reduce the offered load on the network. Once a loss event has occurred or the threshold has been reached, TCP enters the linear growth (congestion avoidance) phase. At this point, the window is increased by 1 segment for each RTT. This happens until a loss event occurs.

Although the strategy is referred to as “slow-start”, its congestion window growth is quite aggressive, more aggressive than the congestion avoidance phase (Jacobson, 1988). Before “slow start” was introduced in TCP, the initial pre-congestion avoidance phase was even faster.

The algorithm begins in the exponential growth phase initially with a congestion window size (cwnd) of 1 or 2 segments and increases it by 1 Segment Size (SS) **for each ACK** received. This behavior effectively doubles the window size each round trip of the network. This behavior continues until the congestion window size (cwnd) reaches the size of the receiver’s advertised window or until a loss occurs.

When a loss occurs half of the current cwnd is saved as a Slow Start Threshold (SSThresh) and slow start begins again from its initial cwnd. Once the cwnd reaches the SSThresh TCP goes into congestion avoidance mode where each ACK increases the cwnd by SS*SS/cwnd. This results in a linear increase of the cwnd.

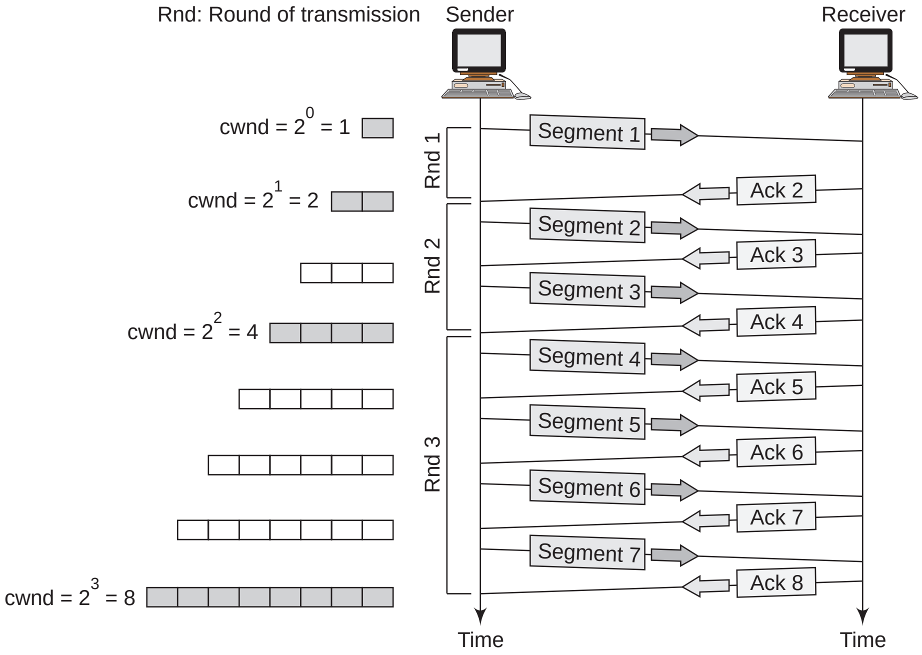


Figure 10.4(a): Congestion control using slow-start algorithm

10.2.2 Additive Increase/Multiplicative Decrease Algorithm

The **additive increase/multiplicative-decrease** (**AIMD**) algorithm is a feedback control algorithm used in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place.

The approach taken is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by 1 MSS (Maximum segment size) every RTT (Round Trip Time) until a loss is detected.

When loss is detected, the policy is changed to be one of multiplicative decrease, which may, for instance, cut the congestion window in half after loss. The result is a saw-tooth behavior that represents the probe for bandwidth.

A loss event is generally described to be either a timeout or the event of receiving 3 duplicate ACKs.

Mathematical Formula

Let *w* be the congestion window; for byte-oriented protocols (such as TCP) the window is relative to the sender's maximum segment size (MSS). Let $a < 1$ and $b\ d \le 1$.

$w \leftarrow a*w$ *decrease when loss is detected*

$w \leftarrow w + b$ *increase when the window has been fully ACKed, or*

$w \leftarrow w + w/b$ *increase by a fraction of MSS when an ACK arrives*

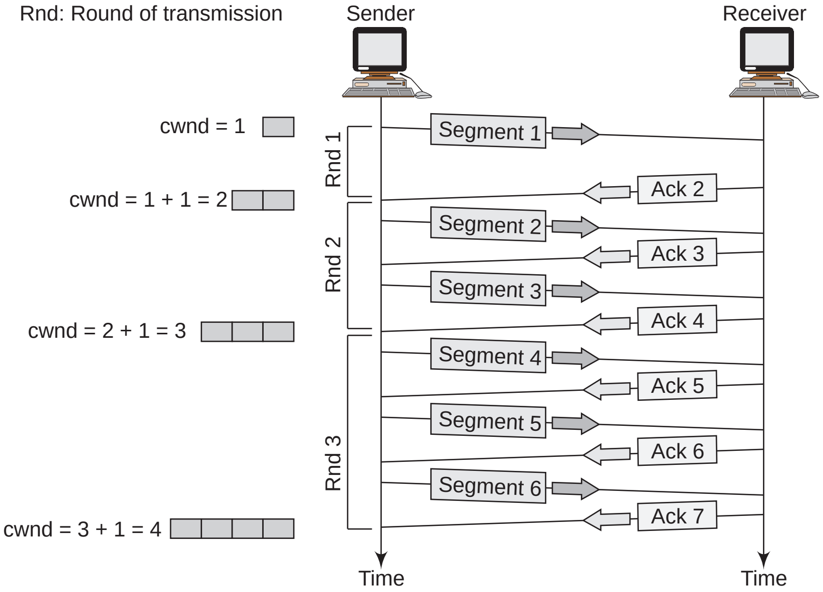


Figure 10.4(b): Congestion control using AIMD algorithm

10.2.3 Congestion Example

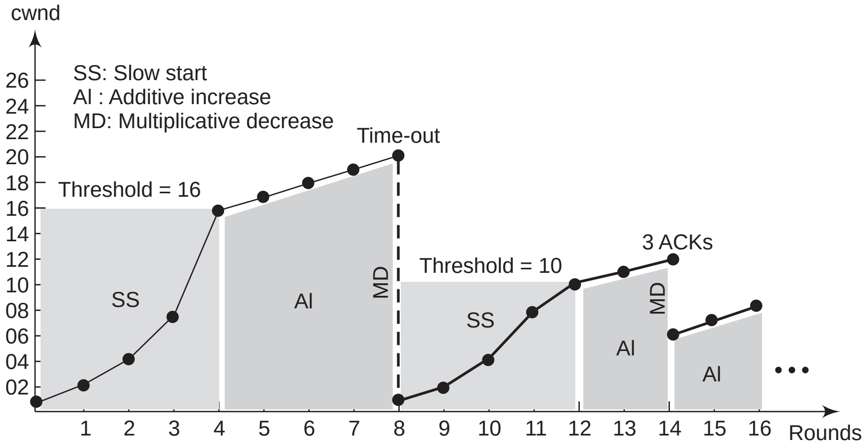


Figure 10.5: Example showing congestion control using multiple methods

10.2.4 Fast Recovery

There is a variation to the slow-start algorithm known as fast recovery, which uses fast retransmit followed by congestion avoidance. In the fast recovery algorithm, during congestion avoidance mode, when packets (detected through 3 duplicate ACKs) are not received, the congestion window size is reduced to the slow-start threshold, rather than the smaller initial value.

Slow-start assumes that unacknowledged segments are due to network congestion. While this is an acceptable assumption for many networks, segments may be lost for other reasons, such as poor data link layer transmission quality. Thus, slow-start can perform poorly in situations with poor reception, such as wireless networks.

The slow-start protocol performs badly for short-lived connections. Older web browsers would create many consecutive short-lived connections to the web server, and would open and close the connection for each file requested. This kept most connections in the slow start mode, which resulted in poor response time. To avoid this problem, modern browsers either open multiple connections simultaneously or reuse one connection for all files requested from a particular web server.

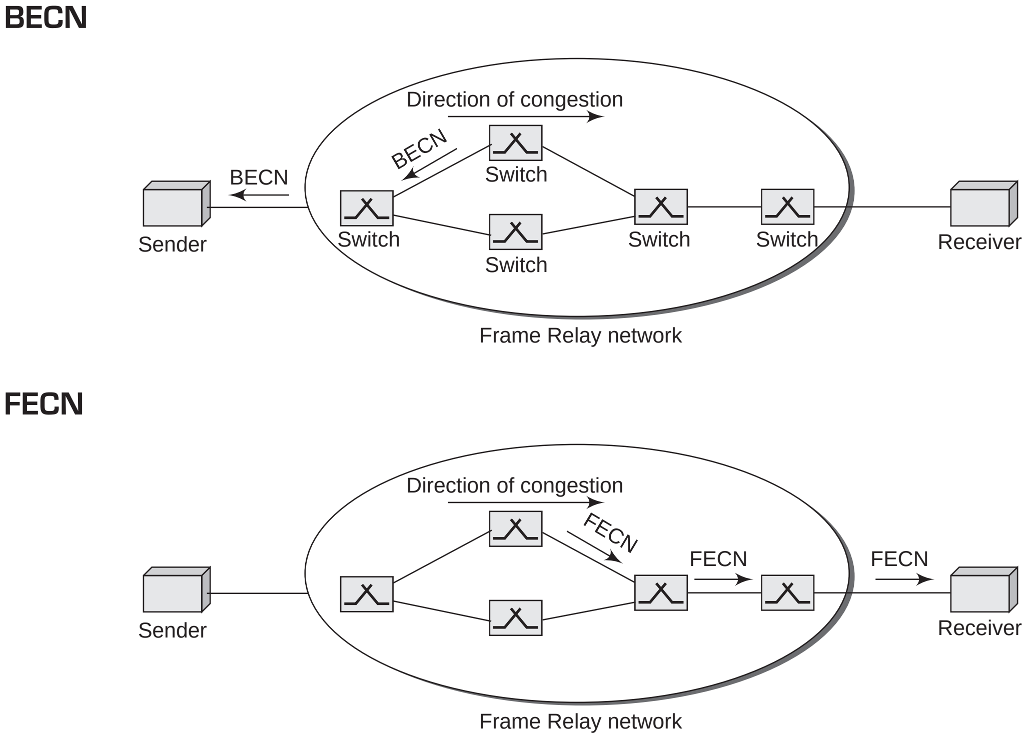
When the network becomes congested to the point that it cannot process new data transmissions, it begins to discard frames. These discarded frames are retransmitted, thus causing more congestion. In an effort to prevent this situation, several mechanisms have been developed to notify user devices at the onset of congestion, so that the offered load may be reduced.

10.3 CONGESTION CONTROL IN FRAME RELAY

Two bits in the **Frame Relay** header are used to signal the user device that congestion is occurring on the line: They are the Forward Explicit Congestion Notification (FECN) bit and the Backward Explicit Congestion Notification (BECN) bit.

The FECN is changed to 1 as a frame is sent downstream toward the destination location when congestion occurs during data transmission. In this way, all downstream nodes and the attached user device learn about congestion on the line.

The **BECN** is changed to 1 in a frame traveling back toward the source of data transmission on a path where congestion is occurring. Thus the source node is notified to slow down transmission until congestion subsides.

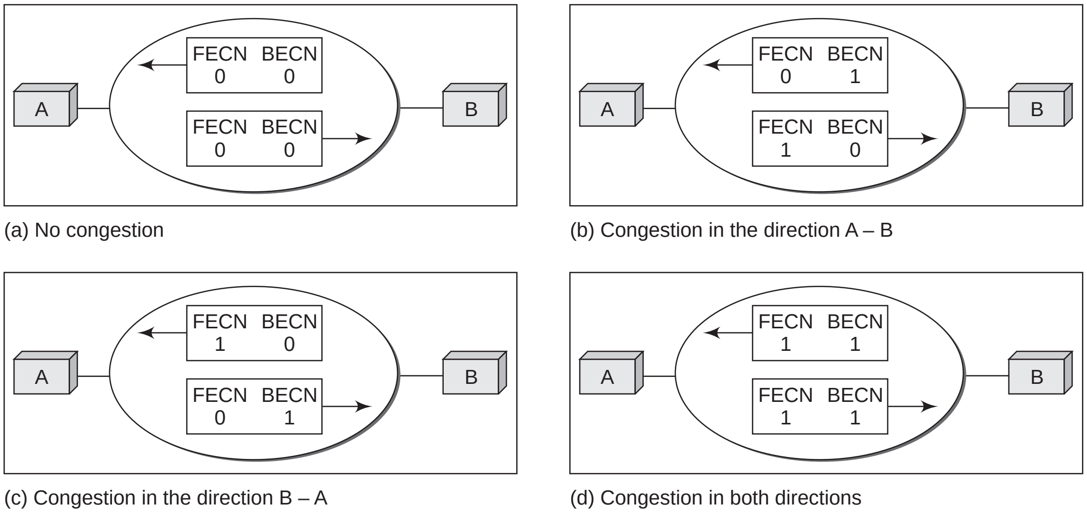


BECN

FECN

Figure 10.6(a): Congestion control in frame relay

Four Cases of Congestion



(a) No congestion

(b) Congestion in the direction A – B

(c) Congestion in the direction B – A

(d) Congestion in both directions

Figure 10.6(b): Different types of congestion

10.4 QUALITY OF SERVICE

Quality of service (*QoS*) refers to resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. For example, a required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed.

Quality of service guarantees are important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as voice over IP, online games and IP-TV, since these often require fixed bit rate and are delay sensitive, and in networks where the capacity is a limited resource, for example in cellular data communication.

A network or protocol that supports QoS may agree on a traffic contract with the application software and reserve capacity in the network nodes, for example during a session establishment phase.

During the session it may monitor the achieved level of performance, for example the data rate and delay, and dynamically control scheduling priorities in the network nodes. It may release the reserved capacity during a tear down phase.

A best-effort network or service does not support quality of service. An alternative to complex QoS control mechanisms is to provide high quality communication over a best-effort network by over-provisioning the capacity so that it is sufficient for the expected peak traffic load. The resulting absence of network congestion eliminates the need for QoS mechanisms.

We briefly discuss four common methods: **Scheduling, Traffic-Shaping, Admission Control** and **Resource Reservation**.

10.4.1 Scheduling

Multiple packets exist in a buffer and they share a common outgoing link. Packet scheduling is necessary when multiple packets compete for a common outgoing link.

10.4.1.1 FIFO Queue

The FIFO Queue which is commonly used in best-effort networks where no QoS guarantees are required, is the simplest scheduling algorithm, simply queues processes in the order that they arrive in the ready queue.

- Throughput can be low, since long packets can hog the network
- Turnaround time, waiting time and response time can be low for the same reasons above
- No prioritization occurs, thus this system has trouble meeting process deadlines.
- The lack of prioritization does permit every process to eventually complete, hence no starvation.

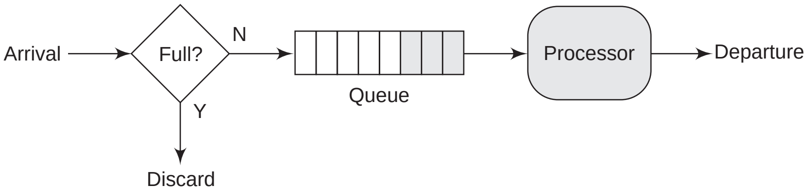


Figure 10.7(a): FIFO queue

To overcome the limitations of FIFO scheduling algorithms, Priority based methods are being used for scheduling of packets across the network. These could be classified, as follows:

10.4.1.2 Classification of Packet Schedulers

- Sorted Priority:
  - o A system potential (a global variable) is updated each time a packet arrives or departs
  - o A timestamp is computed as a function of the system potential for each packet
  - o Packets are sorted and transmitted based on their timestamps
- Frame-based
  - o Time is split into frames of fixed or variable length
  - o Each session makes a reservation in terms of maximum traffic it is allowed to transmit during a frame period
  - o The reservation is made according to a session's allocated rate

Priority Queuing

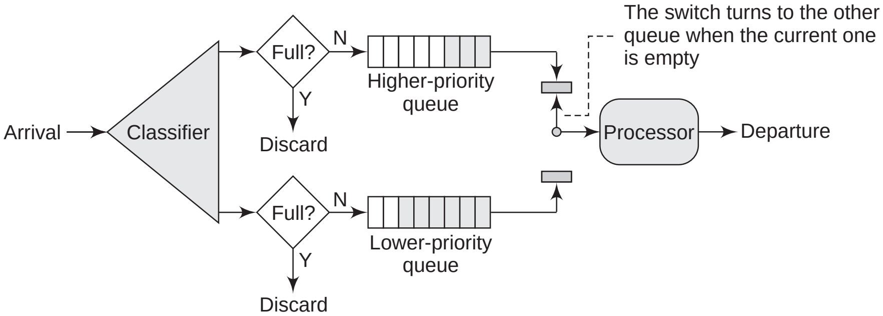


Figure 10.7(b): Priority queuing

Weighted fair queuing (WFQ) is a data packet scheduling technique allowing different scheduling priorities to statistically multiplexed data flows.

WFQ is a generalization of fair queuing (FQ). Both in WFQ and FQ, each data flow has a separate FIFO queue. In FQ, with a link data rate of R, at any given time the N active data flows (the ones with non-empty queues) are serviced simultaneously, each at an average data rate of R / N. Since each data flow has its own queue, an ill-behaved flow (who has sent larger packets or more packets per second than the others since it became active) will only punish itself and not other sessions.

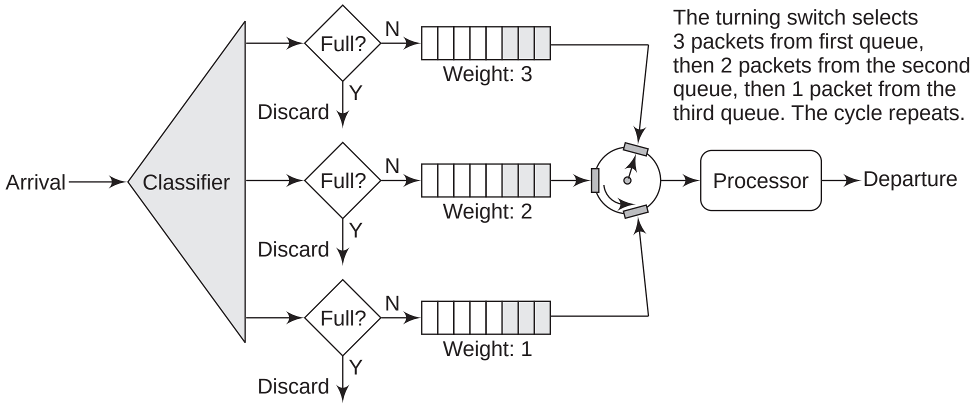

The turning switch selects 3 packets from first queue, then 2 packets from the second queue, then 1 packet from the third queue. The cycle repeats.
Arrival
Classifier
Full?
N
Y
Discard
Weight: 3
Weight: 2
Weight: 1
Processor
Departure

Figure 10.7(c): Weighted fair queueing

10.4.1.3 Frame-based Methods

Round-Robin: In best-effort packet switching and other statistical multiplexing, round-robin scheduling can be used as an alternative to FIFO queuing.

A multiplexer, switch, or router that provides round-robin scheduling has a separate queue for every data flow, where a data flow may be identified by its source and destination address. The algorithm lets every active data flow (that has data packets in the queue) to take turns in transferring packets on a shared channel in a periodically repeated order. The scheduling is work-conserving, meaning that if one flow is out of packets, next data flow will take its place (the scheduling tries to avoid link resources to go unused).

Round-robin scheduling results in *max-min fairness* if the data packets are equally sized, since the data flow that has waited the longest time is given scheduling priority.

Round-robin scheduling may not be desirable if the sizes of the jobs or tasks are strongly varying. A user that produces large jobs would be favored over other users. In that case fair queuing would be preferable.

If guaranteed or differentiated quality of service is offered, and not only best-effort communication, deficit round-robin (DRR) scheduling, weighted round-robin (WRR) scheduling, or weighted fair queuing (WFQ) may be considered.

In multiple-access networks, where several terminals are connected to a shared physical medium, round-robin scheduling may be provided by Token passing channel access schemes such as token ring, as well as by polling or resource reservation from a central control station.

10.4.2 Traffic Shaping

One of the main causes of congestion is that traffic is often bursty. Another open loop method is forcing the packets to be transmitted at a more predictable rate. This method is widely used in ATM networks and is called **traffic shaping**.

When a virtual circuit is set up, the user and the subnet agree on a certain traffic pattern for that circuit.

After reaching an agreement, an issue arises: how can the subnet tell if the user is following the agreement, and what to do if the user is not?

10.4.2.1 Leaky Bucket Algorithm

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant.

This scenario is depicted in figure.

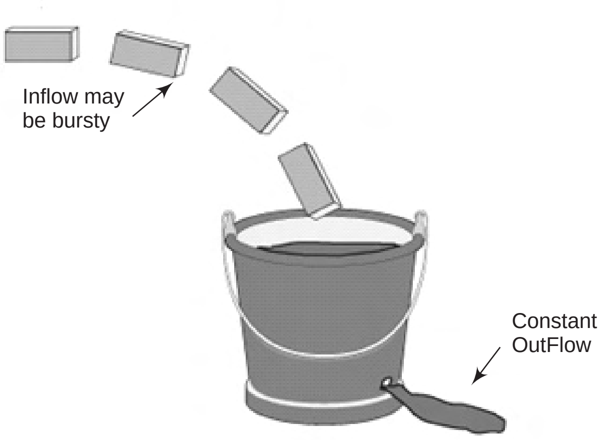


Figure 10.8(a): Uncontrolled flow of packets

Once the bucket is full, any additional water entering it spills over the sides and is lost (i.e. it doesn't appear in the output stream through the hole underneath).

The same idea of leaky bucket can be applied to packets, as shown.

Conceptually each network interface contains a *leaky bucket*. And the following steps are performed:

When the host has to send a packet, the packet is thrown into the bucket.

The bucket leaks at a constant rate, meaning the network interface transmits packets at constant rate.

Bursty traffic is converted to a uniform traffic by the leaky bucket.

In practice the bucket is a finite queue that outputs at a finite rate.

This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.

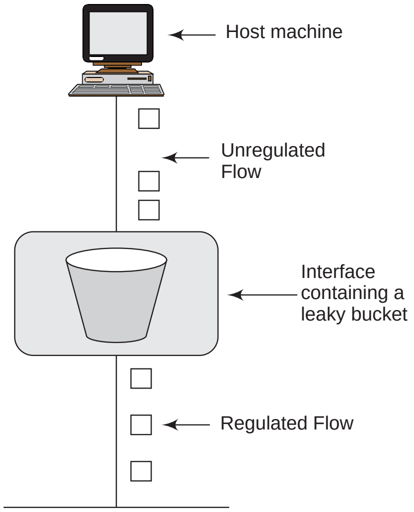


Figure 10.8(b): Flow regulated by leaky bucket algorithm

10.4.2.2 Token Bucket Algorithm

The leaky bucket algorithm described above, enforces a rigid pattern at the output stream, irrespective of the pattern of the input. For many applications it is better to allow the output to speed up somewhat when a larger burst arrives than to loose the data. Token Bucket algorithm provides such a solution. In this algorithm leaky bucket holds token, generated at regular intervals.

Main steps of this algorithm can be described as follows:

- In regular intervals tokens are thrown into the bucket.
- The bucket has a maximum capacity.
- If there is a ready packet, a token is removed from the bucket, and the packet is send.
- If there is no token in the bucket, the packet cannot be send.

The figure below shows the two scenarios before and after the tokens present in the bucket have been consumed. In Figure 10.9(a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface, in Figure 10.9(b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

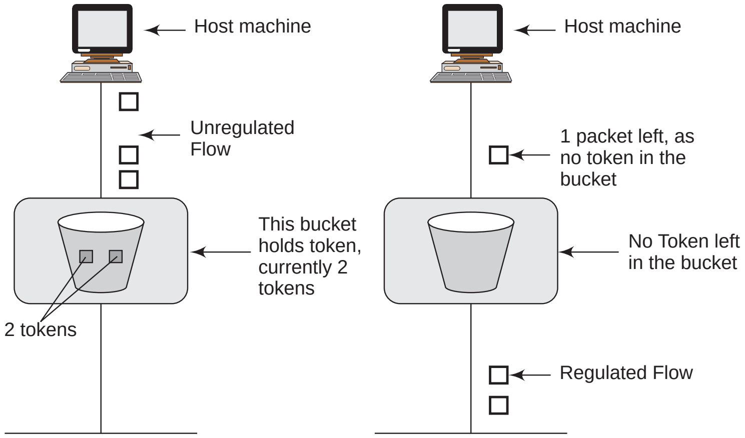

Host machine
Unregulated Flow
This bucket holds token, currently 2 tokens
2 tokens
Host machine
1 packet left, as no token in the bucket
No Token left in the bucket
Regulated Flow

Figure 10.9(a): Token bucket holding two tokens, before packets are sent out Figure 10.9 (b) Token bucket after two packets are sent, one packet still remains as no token is left

The token bucket algorithm is less restrictive than the leaky bucket algorithm, in a sense that it allows bursty traffic. However, the limit of burst is restricted by the number of tokens available in the bucket at a particular instant of time.

10.4.3 QoS Models

The QoS models for the Internet are open standards defined by the IETF.

There are two Internet QoS models currently being standardized within the IETF: *integrated services* and *differentiated services*. These two Internet QoS models augment the traditional best-effort service model described in RFC 1812.

- **Integrated services:** Integrated Services (IS) is a dynamic resource reservation model for the Internet.
- **Differentiated services:** Differentiated Services (DS) removes the per-flow and per-hop scalability issues, replacing them with a simplified mechanism of classifying packets.

10.4.4 Integrated Services

Integrated Services (IS) is a dynamic resource reservation model for the Internet described in RFC 1633.

Hosts use a signaling protocol called Resource ReSerVation Protocol (RSVP) to dynamically request a specific quality of service from the network. QoS parameters are carried in these RSVP messages and each network node along the path installs the parameters to obtain the requested quality of service. These QoS parameters describe one of two currently defined services, guaranteed service and controlled-load service. An important characteristic of IS is that this signaling is done for each traffic flow and reservations are installed at each hop along the route. Although this model is well-suited for meeting the dynamically changing needs of applications, there exist some significant scaling issues that imply it cannot be deployed in a network in which single routers handle many simultaneous flows.

10.4.4.1 Resource Reservation: RSVP

Resource reservation

Here the transmission lines and other devices like bandwidth, buffer space are reserved for the packet transferring

- Another (open loop) strategy relating to virtual circuits is to negotiate an agreement between the host and subnet, so that the subnet can reserve resources along the path when the circuit is set up. Since all necessary resources are guaranteed to be available, congestion is unlikely to occur on the new virtual circuits.
- Reservation can be done all the time, or only when the subnet is congested.

RSVP

The Resource ReSerVation Protocol (RSVP), described in RFC 2205, is a Transport layer protocol designed to reserve resources across a network for an integrated services Internet. RSVP does not transport application data but is rather an Internet control protocol, like ICMP, IGMP, or routing protocols.

RSVP provides receiver-initiated setup of resource reservations for multicast or unicast data flows with scaling and robustness.

RSVP can be used by either hosts or routers to request or deliver specific levels of quality of service (QoS) for application data streams or flows. RSVP defines how applications place reservations and how they can release the reserved resources once the need for them has ended. RSVP operation will generally result in resources being reserved in each node along a path.

RSVP is not itself a routing protocol and was designed to inter-operate with current and future routing protocols.

RSVP by itself is rarely deployed in telecommunications networks today but the traffic engineering extension of RSVP, or RSVP-TE, is becoming more widely accepted nowadays in many QoS-oriented networks.

There are two types of Integrated Services reservations used by the RSVP Agent:

Controlled Load: This reservation type is designed to make the network behave as though it were not loaded, even if one or more of the network elements are experiencing a heavy traffic load.

Guaranteed: This reservation type is designed to allow the network to compute the maximum delay data traffic receives from the network, based on the traffic specification and other known data.

In addition, there are three styles of reservation, depending on how the receiver desires to apply the reservation to its senders:

WF (Wildcard Filter) This style applies a single reservation request to all senders.

FF (Fixed Filter): This style pairs a given reservation request to a given sender. In this way, the receiver can apply a different reservation to each of its senders.

SE (Shared Explicit): This style applies a single reservation to a list of senders. This differs from the WF style in that the list of senders is finite. Additional senders that appear in the future do not automatically inherit an SE style reservation.

Several objects are used in RSVP and RAPI to describe data traffic and reservations. These objects are as follows:

Tspec (traffic specification)

The Tspec is used to describe the sending application data traffic characteristics. It consists of an object known as a token bucket and other related values. A token bucket is a continually sustainable data rate, and the extent to which the rate can exceed the sustainable level for short periods of time.

The Tspec contains these values:

r

Token bucket rate, in bytes per second

b

Token bucket depth, in bytes

p

Peak rate, in bytes per second

m

Minimum policed unit (minimum packet size to be considered), in bytes

M

Maximum packet size (MTU), in bytes

Rspec (guaranteed receiver specification)

An Rspec consists of two values that further describe a reservation request when Guaranteed service is being used:

R

Requested rate, in bytes per second

S

Slack term, in microseconds

Flowspec (reservation specification)

The flowspec is the object used by a receiver application to indicate an actual reservation to be made. The actual makeup of the flowspec depends on the type of reservation. For Controlled Load, the flowspec takes the same form as the sender Tspec (although the form is the same, the receiver might specify different values than the sender). For Guaranteed, the flowspec takes the form of a Tspec followed by an Rspec.

10.4.5 Differentiated Services: DiffServ

This is a computer networking architecture that specifies a simple, scalable and coarse-grained mechanism for classifying, managing network traffic and providing Quality of Service (**QoS**) guarantees on modern IP networks. DiffServ can, for example, be used to provide low-latency to critical network traffic such as voice or video while providing simple best-effort traffic guarantees to non-critical services such as web traffic or file transfers.

DiffServ uses the 6-bit **Differentiated Services Code Point** (**DSCP**) field in the header of IP packets for packet classification purposes. DSCP replaces the outdated IP precedence, a 3-bit field in the Type of Service byte of the IP header originally used to classify and prioritize types of traffic.

DiffServ is intended to address the following difficulties with Intserv and RSVP:

- Scalability: maintaining states by routers in high speed networks is difficult due to the very large number of flows
- Flexible Service Models: Intserv has only two classes, want to provide more qualitative service classes; want to provide 'relative' service distinction (Platinum, Gold, Silver, …)
- Simpler signaling(than RSVP): many applications and users may only want to specify a more qualitative notion of service
- Do fine-grained enforcement only at the edge of the network.
    - o Typically slower links at edges
    - o E.g., mail sorting in post office
- Label packets with a field.
    - o E.g., a priority stamp
- The core of the network uses only the type field for QoS management.
    - o Small number of types with well defined forwarding behavior
    - o Can be handled fast

- Example: expedited service versus best effort
- Evolution rather than revolution

10.4.5.1 Edge Router/Host Functions

- Classification: marks packets according to classification rules to be specified.
- Metering: checks whether the traffic falls within the negotiated profile.
- Marking: marks traffic that falls within profile.
- Conditioning: delays and then forwards, discards or remarks other traffic.

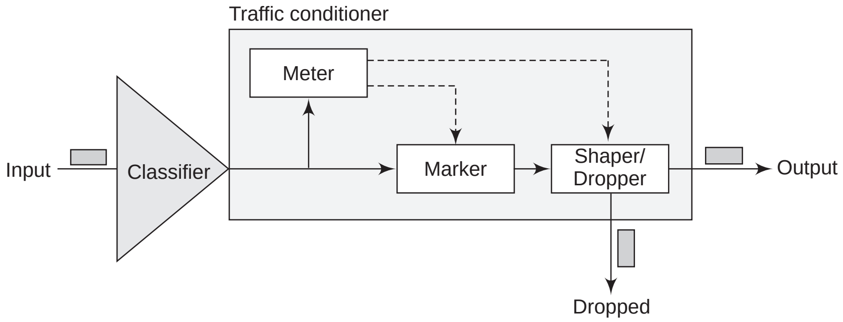


Figure 10.10(a): Edge router and host functions

Classification and Conditioning

- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6.
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive.
- 2 bits are currently unused (CU).

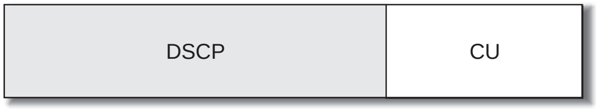


Figure 10.10(b): Classification of packets

Core Functions

- Forwarding: according to "Per-Hop-Behavior" or PHB specified for the particular packet class; such PHB is strictly based on class marking (no other header fields can be used to influence PHB). There is a major advantage, as No state information needs to be maintained by routers.

Forwarding (PHB)

- PHB result in a different observable (measurable) forwarding performance behavior.
- PHB does not specify what mechanisms to use to ensure required PHB performance behavior.
- Examples:
  - o Class A gets x% of outgoing link bandwidth over time intervals of a specified length.
  - o Class A packets leave first before packets from class B.

- **Expedited Forwarding (EF):**
    - o Guarantees a certain minimum rate for the EF traffic.
    - o Implies isolation: guarantee for the EF traffic should not be influenced by the other traffic classes.
    - o Admitted based on peak rate.
    - o Non-conformant traffic is dropped or shaped.
    - o Possible service: providing a virtual wire.
- **Assured Forwarding (AF):**
    - o AF defines 4 classes with some bandwidth and buffers allocated to them.
    - o The intent is that it will be used to implement services that differ relative to each other (*e.g.*, gold, silver,…).
    - o Within each class, there are three drop priorities, which affect which packets will get dropped first if there is congestion.
    - o Lots of studies on how these classes and drop priorities interact with TCP flow control.
    - o Non-conformant traffic is remarked.

Differentiated Services Issues

- **AF and EF are not even in a standard track yet, since research is ongoing.**
- **The key to making Diffserv work is bandwidth management in the network core.**
    - o Simple for simple services such as the virtual pipe, but it is much more challenging for complex service level agreements.
    - o Notion of a "bandwidth broker" that manages the core network bandwidth.
- **Definition of end-to-end services for paths that cross networks with different forwarding behaviors**
    - o Some packets will be handled differently in different routers.
    - o Some routers are not DiffServ capable.

SUMMARY

- The main focus of congestion control and quality of service is data traffic.
- Congestion in a network may occur if the load on the network—the number of packets sent to the network—is greater than the capacity of the network—the number of packets a network can handle.
- In the congestion avoidance algorithm, the size of the congestion window increases additively until congestion is detected.
- Two models have been designed to provide quality of service in the Internet: Integrated Services and Differentiated Services.
- Integrated Services is a flow-based QoS model designed for IP.

11

MULTIMEDIA

Introduction

When we consider the Internet, it is much more than just textual information being distributed using Web Pages. In fact, there is virtually no limit to the types of data we can transfer over the Internet, be it graphics, audio or even video media files. This capacity is in fact possible owing to the confluence of the various media being accepted for transmission over conventional transport options including physical cables as well as wireless, in the form of *multimedia*.

In the following sections, we discuss the possibilities of transferring multimedia content over the Internet, in the form of Streaming audio/video and Interactive audio/video.

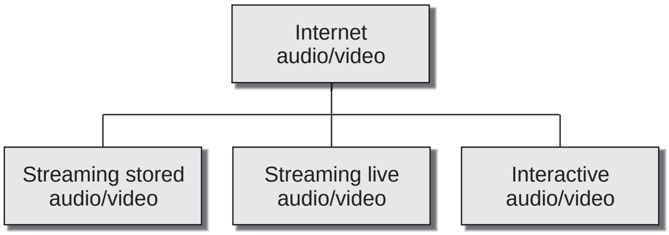


Figure 11.1: Internet audio/video

Streaming stored audio/video refers to on-demand requests for compressed audio/video files. Streaming live audio/video refers to the broadcasting of radio and TV programs through the Internet. Interactive audio/video refers to the use of the Internet for interactive audio/video applications.

11.1 DIGITIZING AUDIO AND VIDEO

Before audio or video signals can be sent on the Internet, they need to be digitized. Moreover, Compression is needed to send audio or video over the Internet. Let us see the methods for audio and video compression.

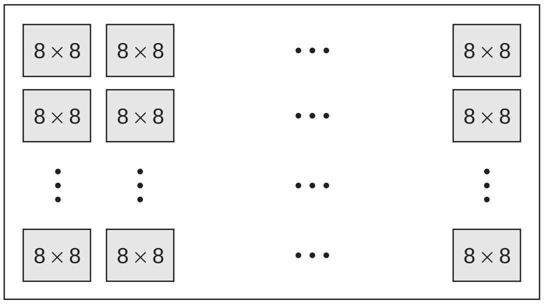


Figure 11.2(a): JPEG gray scale

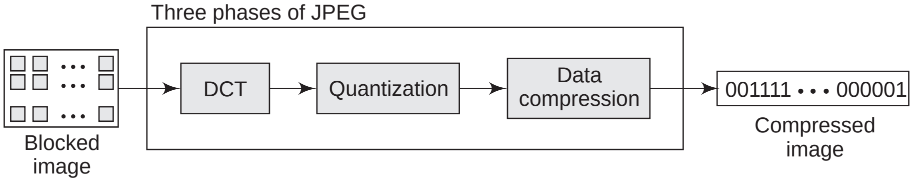
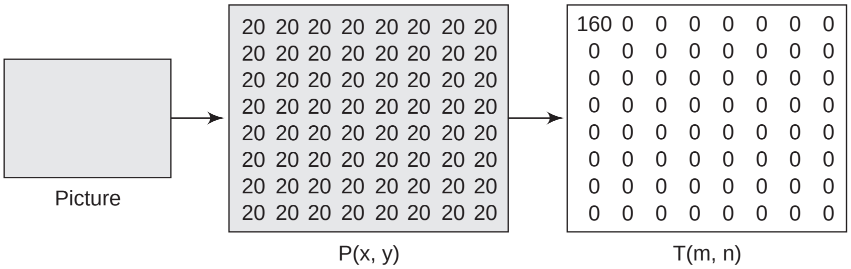


Figure 11.2(b): JPEG process



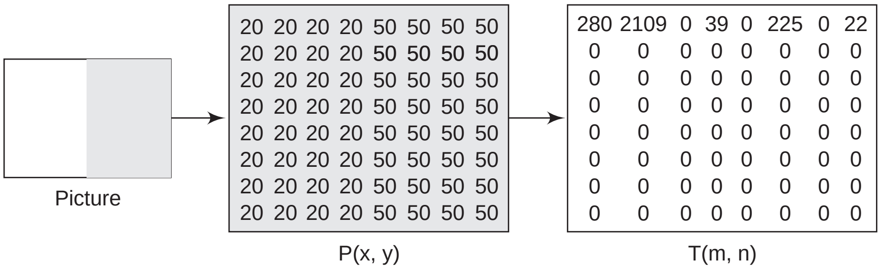
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20

P(x, y)

160	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

T(m, n)

(a) Case 1: uniform gray scale



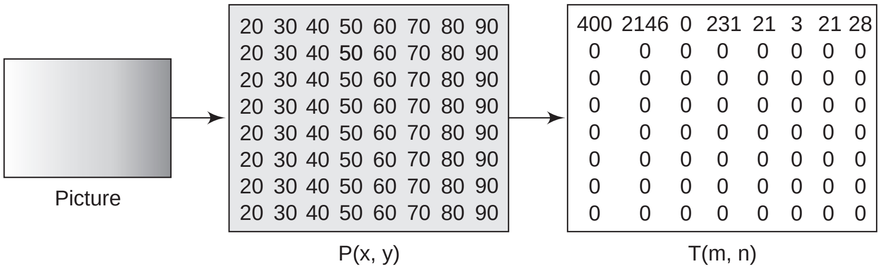
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50
20	20	20	20	50	50	50	50

P(x, y)

280	2109	0	39	0	225	0	22
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

T(m, n)

(b) Case 2: two sections



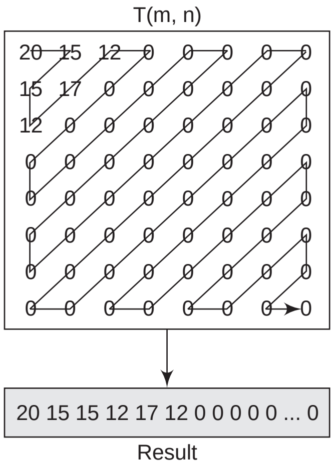
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90
20	30	40	50	60	70	80	90

P(x, y)

400	2146	0	231	21	3	21	28
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

T(m, n)

(c) Case 3: gradient gray scale



20	15	12	0	0	0	0	0
15	17	0	0	0	0	0	0
12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

20 15 15 12 17 12 0 0 0 0 0 ... 0

(d) Reading the table

Figure 11.3: JPEG compression modes

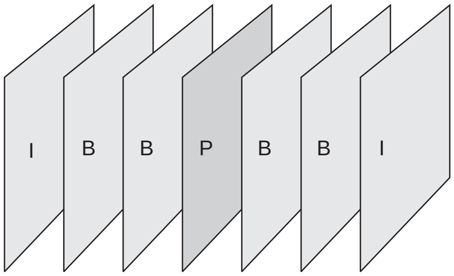


Figure 11.4(a): MPEG frames

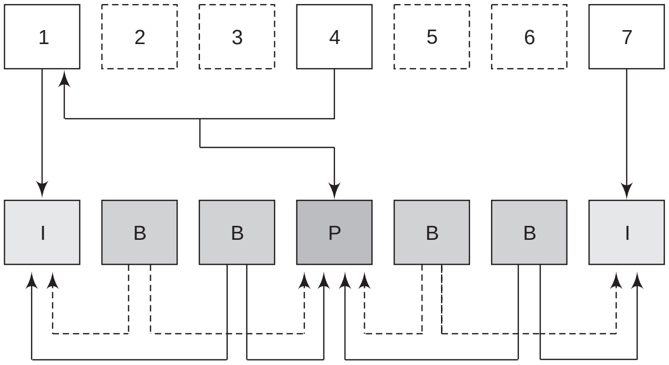
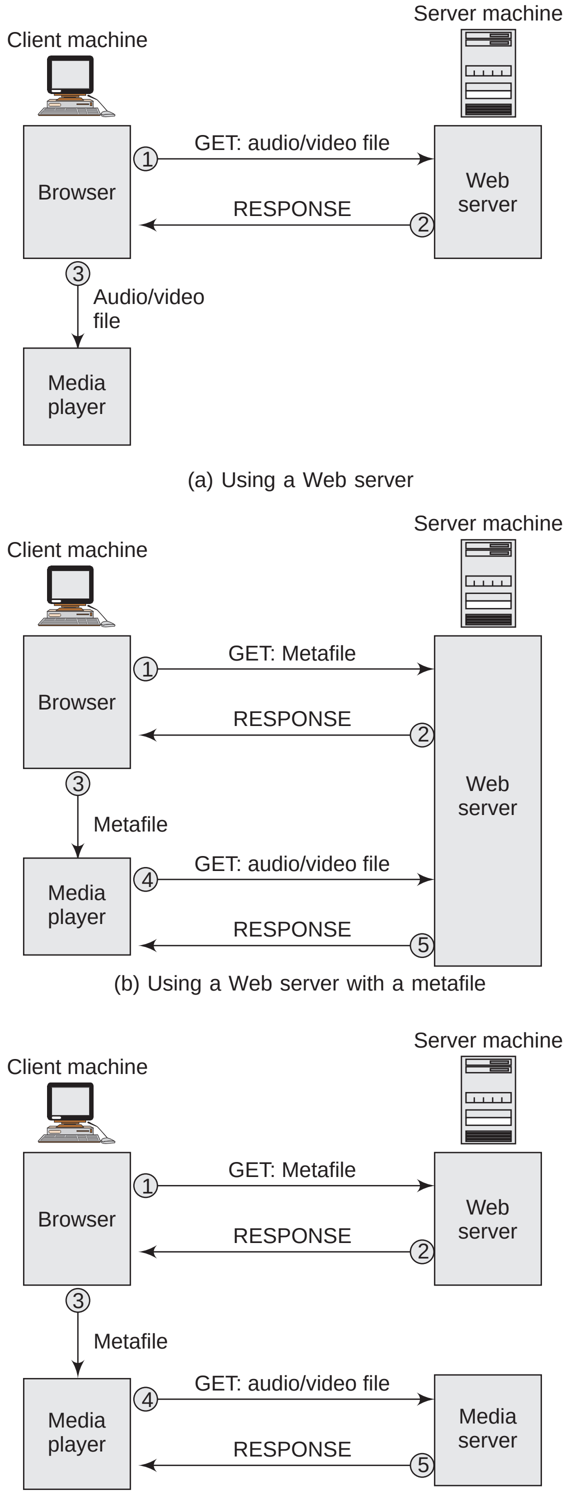


Figure 11.4(b): MPEG frame construction

11.2 STREAMING STORED AUDIO/VIDEO

Now that we have discussed digitizing and compressing audio/video, we turn our attention to specific applications. The first is streaming stored audio and video.

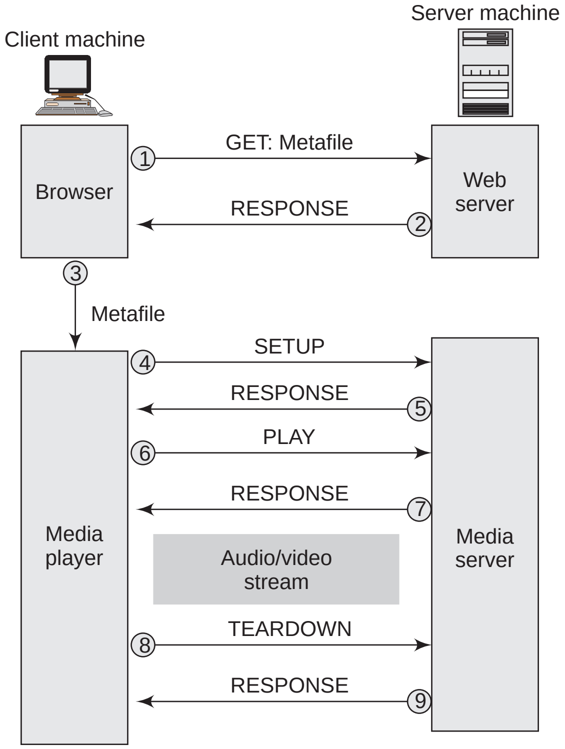
- First Approach: Using a Web Server
- Second Approach: Using a Web Server with a Metafile
- Third Approach: Using a Media Server
- Fourth Approach: Using a Media Server and RTSP



(a) Using a Web server

(b) Using a Web server with a metafile

(c) Using a media server



(d) Using a media server and RTSP

Figure 11.5: Various approaches for streaming stored audio/video

11.3 STREAMING LIVE AUDIO/VIDEO

Streaming live audio/video is similar to the broadcasting of audio and video by radio and TV stations. Instead of broadcasting to the air, the stations broadcast through the Internet. There are several similarities between streaming stored audio/video and streaming live audio/video. They are both sensitive to delay; neither can accept retransmission. However, there is a difference. In the first application, the communication is unicast and on-demand. In the second, the communication is multicast and live.

11.4 REAL-TIME INTERACTIVE AUDIO/VIDEO

In real-time interactive audio/video, people communicate with one another in real time. The Internet phone or voice over IP is an example of this type of application. Video conferencing is another example that allows people to communicate visually and orally.

As the name suggests, real-time interactive audio and video will surely need to be highly time-efficient. A very high level of responsiveness would be desired by the receiver in this case. However, there is every possibility of failure owing to which the interaction could get disrupted.

The most common of these is a *jitter*, which is introduced in real-time data by the delay between packets. To prevent a *jitter* from occurring, we can *time-stamp* the packets and separate the *arrival time* from the *playback time*.

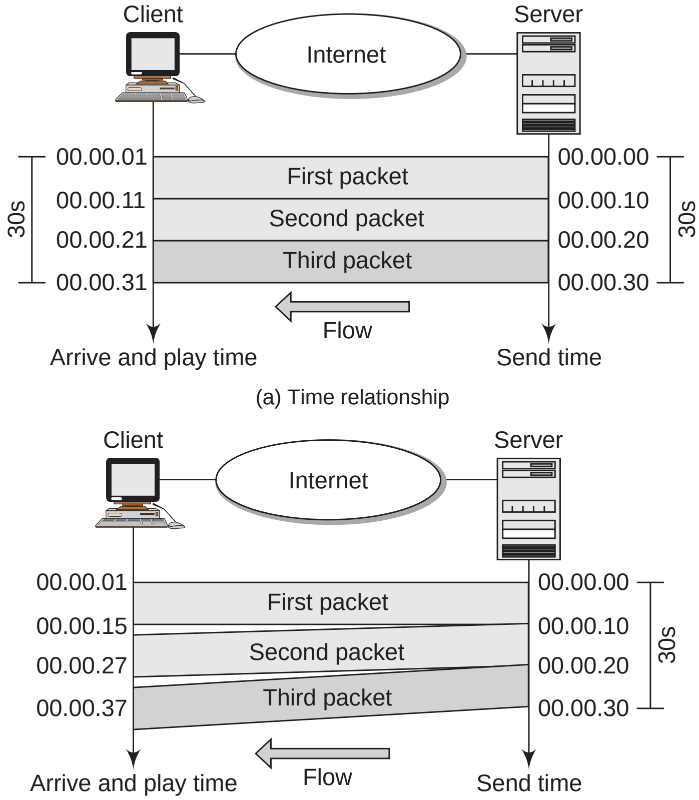
Thus, ***real-time*** traffic can be made possible by using a mechanism called the ***Playback buffer***. A playback buffer would provide a intermediary space, where the entire message could be stored before being forwarded to the recipient, so that the receiver experiences the same effect that a live-video would give, without feeling the time difference, as it would be just a matter of a few seconds, to the maximum.

In order to assure that the message would be proper when forwarded, there would be a sequence number on each packet. Moreover, as a unicast mode would make the receiver dependent solely on the sender, it would be more desirable to support *multicasting* in the case of interactive audio/video.

Another common issue that can lead to a failure in interactive systems is the difference in bandwidth on the sender and receiver side. However, this problem can be solved by the means of **Translation**, i.e., changing the encoding of a payload to a lower quality to match the bandwidth of the receiving network.

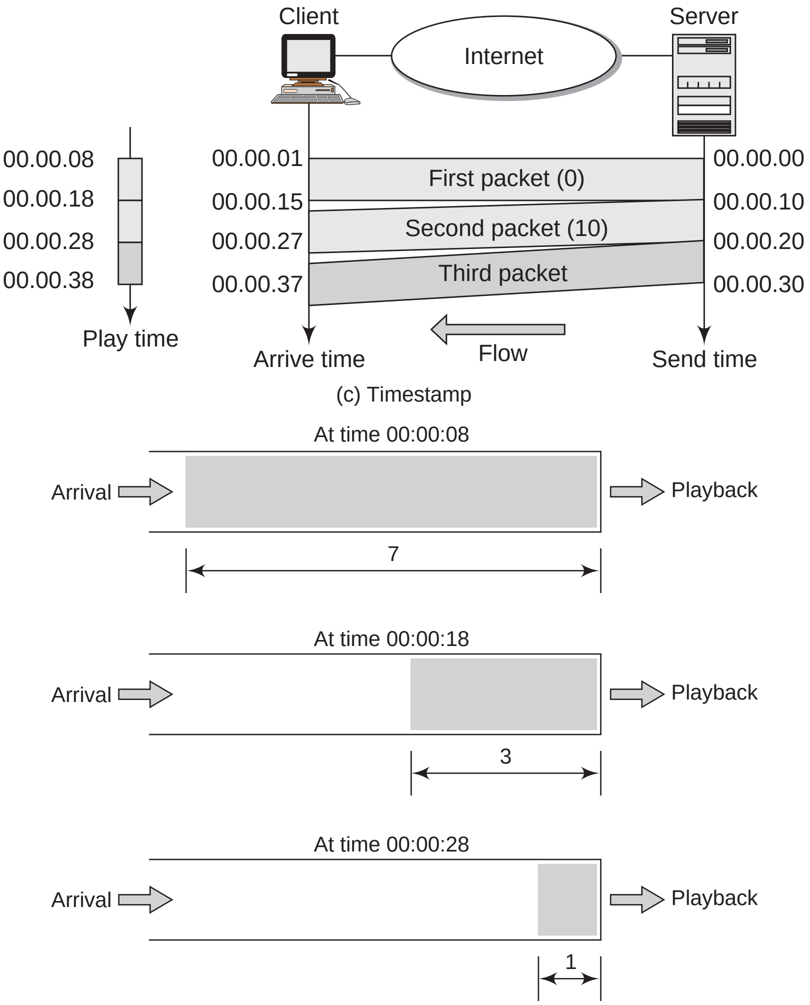
We can also minimize wastage of bandwidth by Mixing of signals; which means combining several streams of traffic into one stream.

The following is a graphical description of the entire process of Multimedia communication and its characteristics:



(a) Time relationship

(b) Jitter



(c) Timestamp

(d) Playback buffer

Figure 11.6: Characteristics of multimedia communication

11.5 PROTOCOLS FOR SUPPORTING INTERACTIVE MESSAGES

TCP, even though it is more advanced, is not suitable for interactive multimedia traffic because we cannot allow retransmission of packets. Thus, UDP is more suitable than TCP for interactive traffic. However, we need the services of RTP, another transport layer protocol, to make up for the deficiencies of UDP.

11.5.1 RTP

Real-time Transport Protocol (RTP) is the protocol designed to handle real-time traffic on the Internet. RTP does not have a delivery mechanism; it must be used with UDP. RTP stands between UDP and the application program. The main contributions of RTP are *time-stamping*, *sequencing*, and *mixing* facilities, which as we saw above are the essential characteristics of any interactive message.

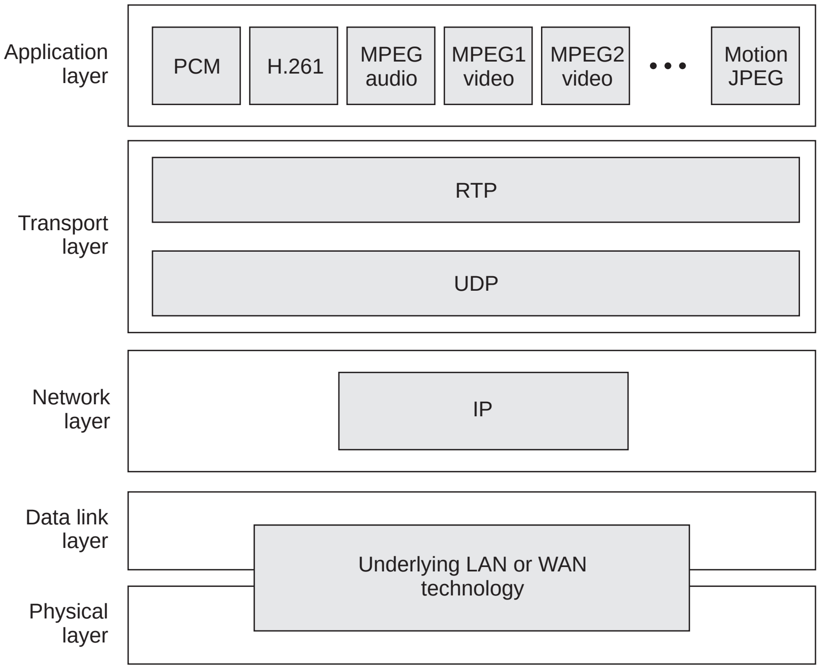


Figure 11.7(a): RTP as a transport protocol

Following the *connectionless* mode, RTP uses a temporary, even-numbered UDP port in order to perform message transfer. Let us now see the *header format* and types of *payload* for an RTP packet:

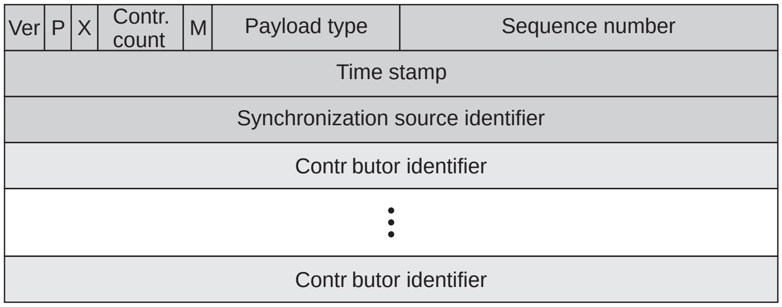


Figure 11.7(b): RTP packet header format

Payload Types

Table 11.1 RTP payload types

Type	Application	Type	Application	Type	Application
0	PCMμ Audio	7	LPC audio	15	G728 audio
1	1016	8	PCMA audio	26	Motion JPEG
2	G721 audio	9	G722 audio	31	H.261
3	GSM audio	10-11	L16 audio	32	MPEG1 video
5-6	DV14 audio	14	MPEG audio	33	MPEG2 video

There is, however, one major disadvantage with RTP that it allows only messages containing data from the source to the destination. There may be the need to transfer many other types of messages such as flow control or feedback information. This drawback was addressed by another protocol, RTCP.

11.5.2 RTCP

RTCP allows additional messages to be transferred that can control the flow and quality of data and also allow the recipient to send feedback to the source (or sources). RTCP uses an Odd numbered port that follows the port number selected for RTP. The following diagram depicts the different message types for RTCP:

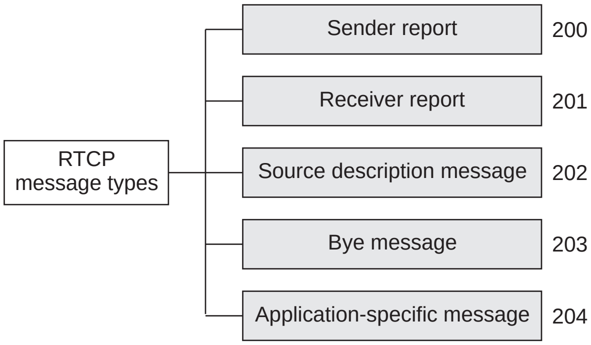


Figure 11.8: RTCP message types

We consider here the most common example of a real-time interactive audio/video application: VoIP; i.e. Voice over IP also known as Internet Telephony.

11.5.3 VoIP

This technology aims to use the Internet as a *telephone* network, with some additional capabilities. Two protocols have been designed to handle this type of communication: **SIP** and **H.323**.

We will first see the mechanism used by SIP, followed by H.323.

The Session Initiation Protocol (SIP) is one of the most important VoIP signaling protocols operating in the application layer in the five-layer TCP/IP model. SIP can perform both unicast and multicast sessions and supports user mobility.

SIP handles signals and identifies user location, call setup, call termination, and busy signals. SIP can use multicast to support conference calls and uses the Session Description Protocol (SDP) to negotiate parameters.

11.5.3.1 SIP Components

The following figure shows an overview of SIP. A call is initiated from a user agent: the user's IP telephone system, which is similar to a conventional phone. A user agent assists in initiating or terminating a phone call in VoIP networks. A user agent can be implemented in a standard telephone or in a laptop with a microphone that runs some software. A user agent is identified using its associated domain. For example, user1@domain1.com refers to user 1, who is associated with the domain1.com network.

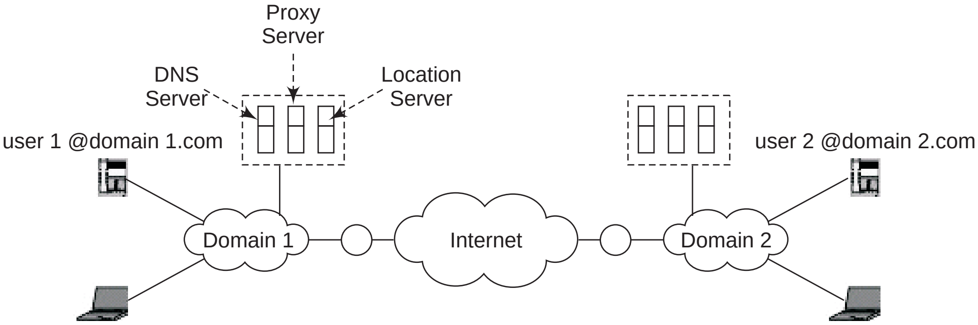


Figure 11.9: Overview of SIP

SIP consists of the following five servers:

1. DNS server. The Domain Name System (DNS) server maps the domain name to an IP address in the user information database (UID). The UID database contains such user information as preferences and the services to which it has subscribed. The UID also stores information on the related IP addresses. Each user is normally configured with more than one DNS server.
2. Proxy server. The proxy server forwards requests from a user agent to a different location and handles authorizations by checking whether the caller is authorized to make a particular call.
3. Location server. This server is responsible for UID management. The location server interacts with the database during call setup. Each proxy server is normally configured with more than one location server.
4. Redirect server. This server performs call forwarding and provides alternative paths for the user agent.
5. Registrar server. This server is responsible for registering users in the system and updating the UID that the location server consults. Requests for registration must be authenticated before the registration is granted.

The following are the different types of ***SIP messages*** and ***SIP formats***:

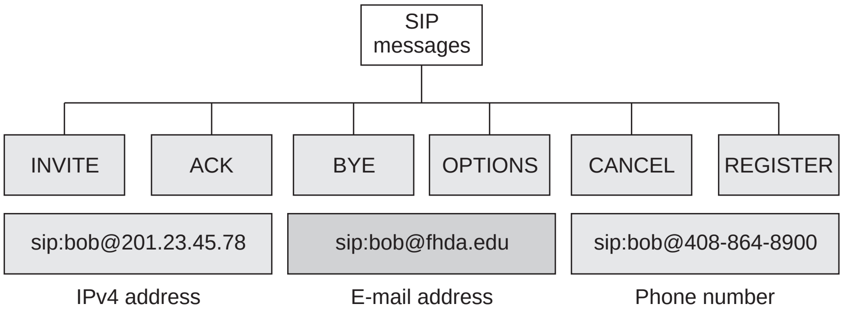
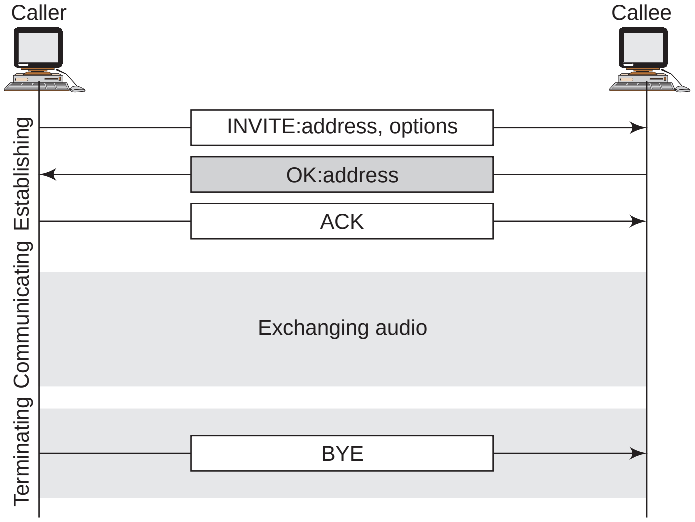


Figure 11.10: SIP messages and formats

Let us see a descriptive example of the procedure for the establishment and teardown of a SIP session:



The following is a description of the procedures used by SIP to ***Track the callee***:

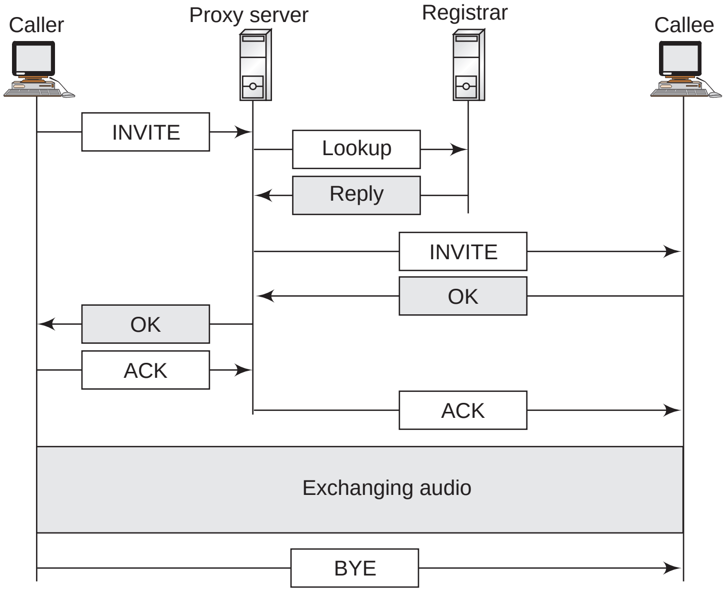


Figure 11.11: Establishment and teardown of SIP session

11.5.3.2 H.323

The H.323-series protocols are implemented in layer 5 of the TCP/IP model and run over either TCP or UDP. The H.323 protocols interact to provide ideal telephone communication, providing phone numbers to IP address mapping, handling digitized audio streaming in IP telephony, and providing signaling functions for call setup and call management. The H.323 series supports simultaneous voice and data transmission and can transmit binary messages that are encoded using basic encoding rules. From the security standpoint, the H.323 scheme provides a unique framework for security, user authentication, and authorization and supports conference calls and multipoint connections, as well as accounting and call-forwarding services. The following diagram describes the architecture followed by the H.323 protocol.

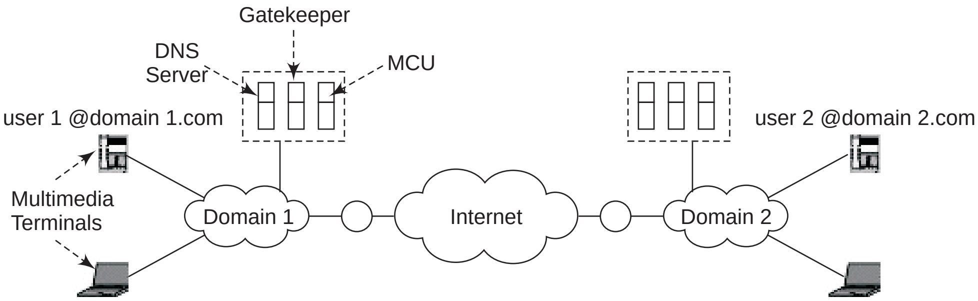


Figure 11.12: Use of gatekeepers

A gatekeeper can send signals to other gatekeepers of different zones to access users of those domains. This feature supports distributed locations of multinational systems.

H.323 protocols

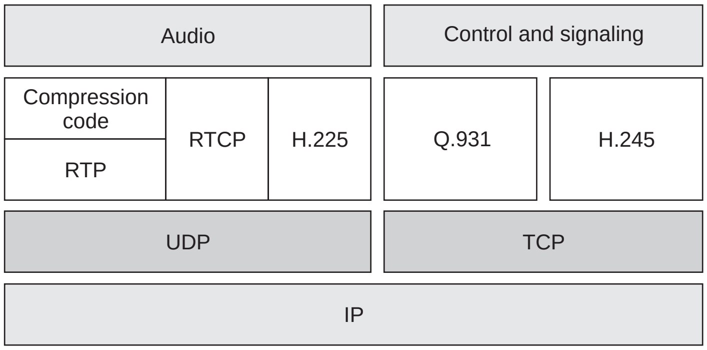


Figure 11.13: H.323 protocol stack

The following is an example of an implementation of the H.323 protocol:

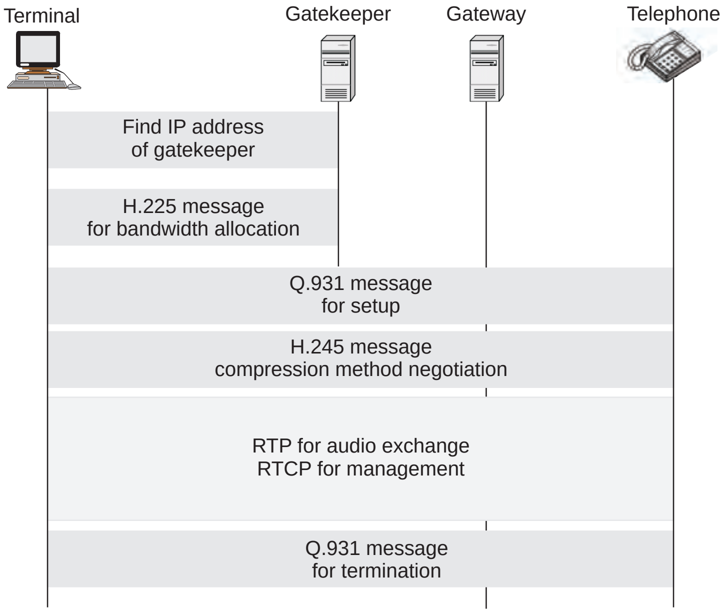

Terminal
Gatekeeper
Gateway
Telephone
Find IP address of gatekeeper
H.225 message for bandwidth allocation
Q.931 message for setup
H.245 message compression method negotiation
RTP for audio exchange
RTCP for management
Q.931 message for termination

Figure 11.14: Implementation of H.323 protocol

Session Signaling and Numbering

The following figures show the details of two IP telephone communications using H.323-series protocols. Assume that two user agents 1 and 2: user1@domain1.com and user2@domain2.com, respectively. In this example, user 1 places a call to contact user 2.

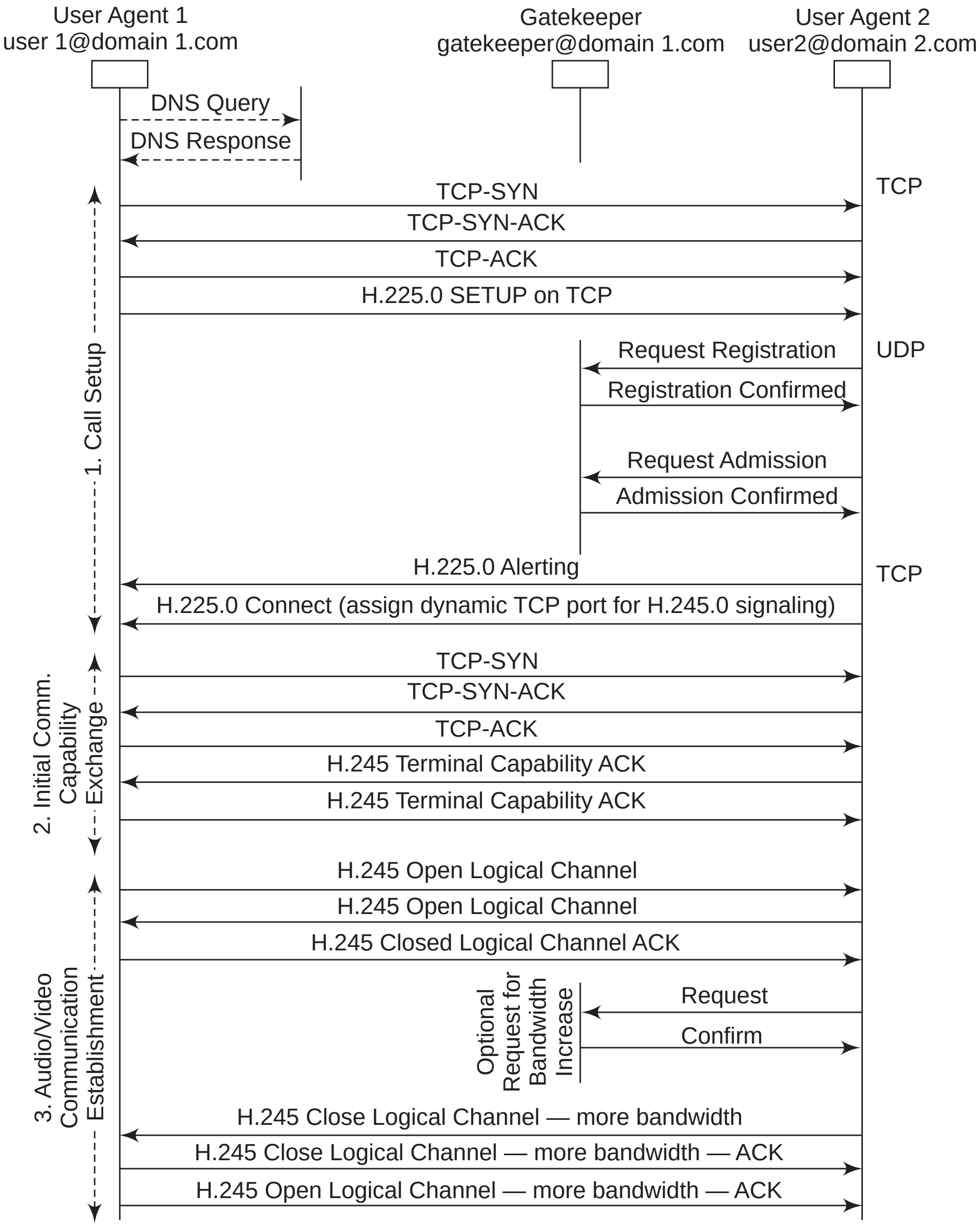

User Agent 1
user 1@domain 1.com
Gatekeeper
gatekeeper@domain 1.com
User Agent 2
user2@domain 2.com
DNS Query
DNS Response
TCP-SYN
TCP
TCP-SYN-ACK
TCP-ACK
H.225.0 SETUP on TCP
1. Call Setup
Request Registration
UDP
Registration Confirmed
Request Admission
Admission Confirmed
H.225.0 Alerting
TCP
H.225.0 Connect (assign dynamic TCP port for H.245.0 signaling)
2. Initial Comm. Capability Exchange
TCP-SYN
TCP-SYN-ACK
TCP-ACK
H.245 Terminal Capability ACK
H.245 Terminal Capability ACK
3. Audio/Video Communication Establishment
H.245 Open Logical Channel
H.245 Open Logical Channel
H.245 Closed Logical Channel ACK
Optional Request for Bandwidth Increase
Request
Confirm
H.245 Close Logical Channel — more bandwidth
H.245 Close Logical Channel — more bandwidth — ACK
H.245 Open Logical Channel — more bandwidth — ACK

Figure 11.15(a): H.323 protocol signaling: steps 1, 2, and 3

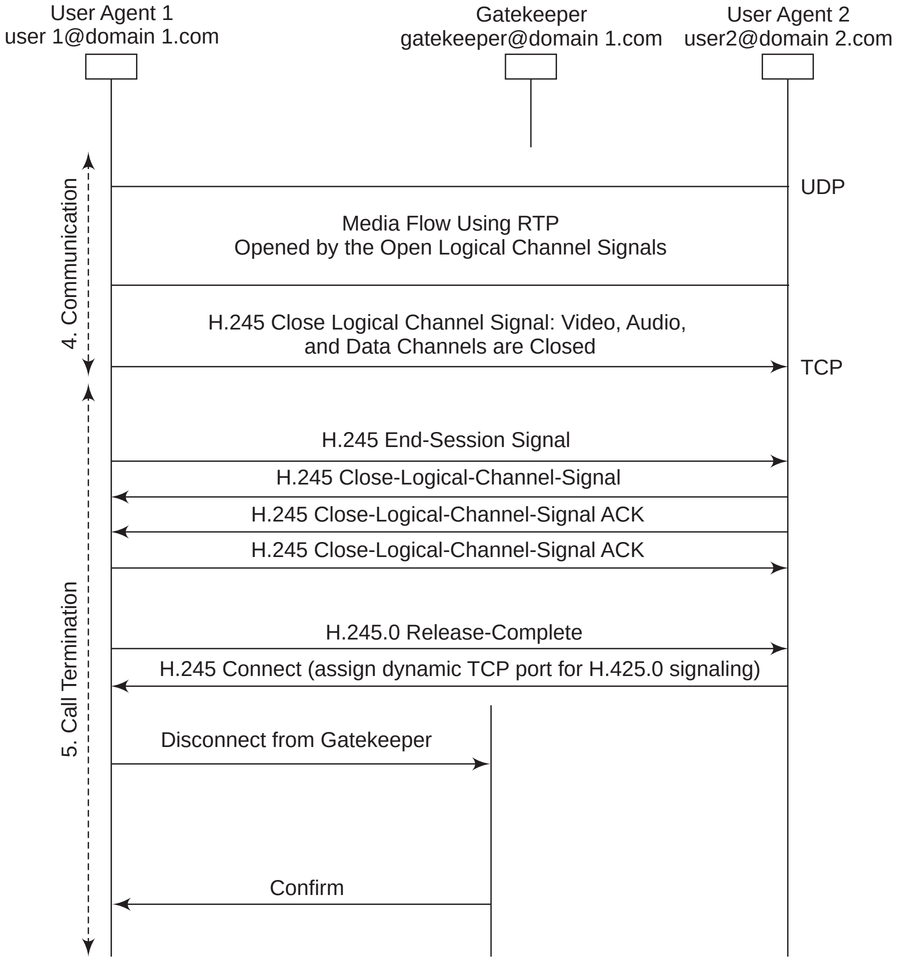

User Agent 1
user 1@domain 1.com
Gatekeeper
gatekeeper@domain 1.com
User Agent 2
user2@domain 2.com
UDP
Media Flow Using RTP
Opened by the Open Logical Channel Signals
H.245 Close Logical Channel Signal: Video, Audio, and Data Channels are Closed
TCP
4. Communication
H.245 End-Session Signal
H.245 Close-Logical-Channel-Signal
H.245 Close-Logical-Channel-Signal ACK
H.245 Close-Logical-Channel-Signal ACK
H.245.0 Release-Complete
H.245 Connect (assign dynamic TCP port for H.425.0 signaling)
Disconnect from Gatekeeper
Confirm
5. Call Termination

Figure 11.15(b): H.323 protocol signaling: steps 4 and 5

11.5.3.3 Communication Methods

First, user 1 communicates with its DNS server, as explained earlier. The signaling uses both UDP and TCP, and partitioned into the following five steps:

1. Call setup
2. Initial communication capability
3. Audio/video communication establishment
4. Communications
5. Call termination

At step 1, when user1 dials user 2's telephone number, the first set of signals are exchanged between these two users in conjunction with opening a TCP connection. TCP-SYN, TCP-SYN-ACK, and then TCP-ACK signals are generated between the two users. At this point, the H225.0 SETUP ON TCP signal informs the called party that the connection can be set up on TCP. The users can now request a certain bandwidth from the associated gatekeeper server of the called party. The requested bandwidth is granted if sufficient bandwidth is left over on the connection; otherwise, the call has to find another gatekeeper to register with. This phase is handled by H.225 and H.245 protocols.

At step 2, all the end points' communication capabilities available over TCP are exchanged. This phase is necessary because the type of communication service requested depends on the communication capabilities of both end points. Step 3 implements the establishment of a logical channel, which in H.323 is unidirectional; therefore, a logical channel must be established in either direction in order to have two-way communications. At the end of this phase, two end points are set for communications. Meanwhile, more bandwidth can also be requested, as shown in the figure, before communications start.

Step 4 comprises the communications between the two users. This phase is handled using RTP over UDP. At this step, any kind of media flow can be considered, depending on the size and type of the channel established in step 4. Finally at step 5, the call is terminated by either user. In the above figure, user 2 initiates the termination of the call by closing the logical channel in H.245 and disconnecting the call from the gatekeeper.

SUMMARY

- Streaming stored audio/video refers to on-demand requests for compressed audio/video files.
- Streaming live audio/video refers to the broadcasting of radio and TV programs through the Internet.
- Interactive audio/video refers to the use of the Internet for interactive audio/video applications.
- Compression is needed to send video over the Internet.
- Jitter is introduced in real-time data by the delay between packets.
- To prevent jitter, we can time-stamp the packets and separate the arrival time from the playback time.
- A sequence number on each packet is required for real-time traffic.

12

WIRELESS AD-HOC NETWORKS

Introduction

Mobile ad-hoc networks (MANETs) have had a huge impact in the world of computer networks. Mainly identified by the unbound establishment of a wireless network wherever and whenever needed, the MANET infrastructure supports several *location-independent* services.

Ad-hoc networks do not need any fixed infrastructure to operate and thus can support dynamic topology scenarios where wired infrastructure do not exist. However, the concerns of Routing and Security persist even for ad-hoc networks.

A mobile user can act as a routing node, and a packet can be routed from a source to its destination without having any static router in the network. Two classes of routing strategies in ad-hoc networks are *table-driven* routing protocols and *source-initiated* routing protocols.

Security of ad-hoc networks is a key issue. Ad-hoc networks are basically prone to attacks as an intruder can easily attack ad-hoc networks by loading available network resources and disturbing the normal operation of routing protocols by simply adding modifying packets.

12.1 OVERVIEW OF WIRELESS AD-HOC NETWORKS

Wireless mobile ad-hoc network (MANET) technology is designed for the establishment of a network anywhere and anytime, without any fixed infrastructure to support the mobility of the users in the network. In other words, a wireless ad-hoc network is a collection of mobile nodes with a dynamic network infrastructure forming a temporary network. Such networks no central server or base station for providing connectivity and all network intelligence must be placed inside the mobile user devices.

Following is a descriptive overview of an ad-hoc network, where wireless mobile nodes have formed a network, with one node too far to reach.

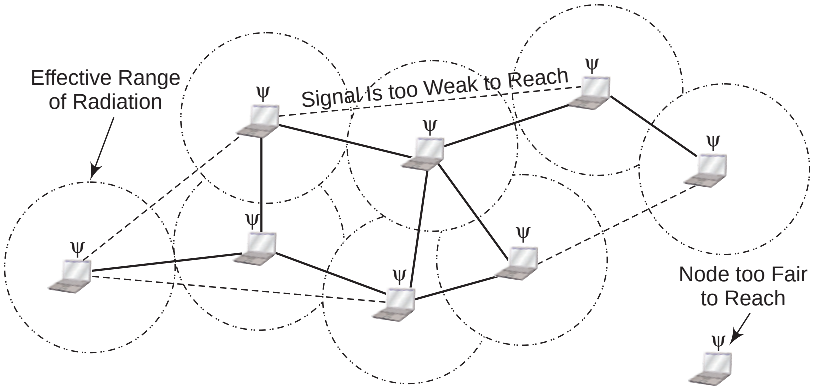


Figure 12.1: Overview of an ad-hoc network

In such an environment, each mobile user acts as a routing node, and a packet is routed from a source to its destination by incorporating of other network users. Since the topology of an ad-hoc network can change quickly and unpredictably, it should be adaptable to changes, such as when a link breaks, a node leaves the network, or a new node is attached to the network. Thus, unlike intra-domain routing algorithm for regular networks, if a node leaves an ad-hoc network, all affected nodes can discover new routes.

12.2 APPLICATIONS OF AD-HOC NETWORKS

Ad-hoc networks have several types of applications:

- **Rescue operations**: In an emergency public disaster, such as an earthquake, ad-hoc networks can be set up at a location where no infrastructure is present. Ad-hoc networks can be used to support network connectivity when no fixed network is available.
- **Military**: Ad-hoc networks can be used in a battle zone, for a military command and mobile units.
- **Law and Security**: An ad-hoc network can be used in a temporary security operation, acting as a mobile surveillance network, to ensure law and order.
- **Home networks**: An ad-hoc network can be used to support seamless connectivity among various devices.
- **Conferencing**: Ad-hoc networks can be set up for a presentation. An audience can download a presentation, browse the slides on a portable device, print them on the local printer, or e-mail the presentation to an absent colleague.

12.3 FEATURES OF AD-HOC NETWORKS

Ad-hoc networks must possess several unique features. One is ***automatic discovery*** of available services. Each time a new service becomes available, an ad hoc networking device has to configure use of the new service. As an ad-hoc network lacks centralized administration, the network must be able to prevent network collapse when one of the mobile nodes moves out of transmitter range.

In general, nodes should be able to enter or leave the network as they wish. Thus, every node acts as both a host and a router, and the network must be intelligent enough to handle network dynamics. This property is called ***self-stabilization***.

One of the most common tasks of an ad-hoc network is to ***multicast a message*** to many users efficiently. In such an environment, networks are subject to severe blocking. Thus, the performance of an ad hoc system depends on the stability of the network architecture. The inclusion of all these features in ad-hoc networks requires considerable architecture sophistication.

12.4 ROUTING IN AD-HOC NETWORKS

Routing of packets is quite difficult in ad-hoc networks owing to the lack of a backbone infrastructure. A routing protocol should be able to automatically recover from any problem in a finite amount of time without human intervention.

Conventional routing protocols are designed for non-moving infrastructures and assume that routes are bidirectional, which is not always the case for ad-hoc networks. Identification of mobile terminals and correct routing of packets to and from each terminal while moving are certainly challenging.

Since the topology of an ad-hoc network is dynamic, reserving resources and sustaining QoS are difficult. In an ad-hoc medium, it may not be possible to communicate in both directions so ad-hoc routing protocols should assume that links are unidirectional.

The *power* of a wireless device is another important factor. The routing protocol also has to support node stand-by mode. Devices such as laptops are very limited in battery power; hence, the use of stand-by mode to save power is important.

12.4.1 Classification of Routing Protocols

Ad-hoc routing protocols can be classified into two broad categories:

1. **Based on Location**: Centralized and Distributed.

   In centralized routing protocols, the routing decision is made at a central node. In distributed routing protocols, the routing decision is made by all the network nodes. Routing protocols in most efficiently designed ad-hoc networks are distributed to increase the reliability of the network. In such cases, nodes can enter or leave the network easily, and each node can make routing decisions, using other collaborative nodes.
2. **Based on Dynamics**: Static and Adaptive.

   In static routing protocols, a route of a source/destination pair does not change because of any traffic condition or link failure. In adaptive routing protocols, routes may change because of any congestion.

12.4.2 Routing Protocols for Ad-Hoc Networks

Whether a protocol is centralized, distributed, static, or adaptive, it can generally be categorized as either ***table driven*** or ***source initiated***.

12.4.2.1 Table-Driven Routing Protocols

Table-driven, or proactive, routing protocols find routes to all possible destinations ahead of time. The routes are recorded in the routing tables at the nodes and are updated within the predefined intervals.

Proactive protocols are faster in decision-making but need more time to converge to a steady state, causing problems if the topology of the network continually changes. However, maintaining routes can lead to a large overhead.

Table-driven protocols require every node to maintain one or more tables to store updated routing information from every node to all other nodes. Nodes propagate updated tables all over the network such that the routing information in each table corresponds to topological changes in the network.

12.4.2.2 Source-Initiated Routing Protocols

Source-initiated, or reactive, routing protocols are *on-demand* procedures and create routes only when requested to do so by source nodes. A route request initiates a route-discovery process in the network and is completed once a route is discovered. If it exists at the time of a request, a route is maintained by a route-maintenance procedure until either the destination node becomes irrelevant to the source or the route is no longer needed.

On-demand protocols are more suitable for ad-hoc networks. In most cases, reactive protocols are desired. This means that the network reacts only when needed and does not broadcast information periodically. However, the control overhead of packets is smaller than for proactive protocols.

We now describe three table-driven protocols and four source-initiated protocols. The table-driven protocols include the Destination-Sequenced Distance Vector (DSDV) protocol, the Cluster-Head Gateway Switch Routing (CGSR) protocol, and the Wireless Routing Protocol (WRP).

The source-initiated protocols are the Dynamic Source Routing (DSR) protocol, the Associative-Based Routing (ABR) protocol, Temporally Ordered Routing Algorithm (TORA), and Ad-Hoc On-Demand Distance Vector (AODV) protocol.

12.5 TABLE DRIVEN ROUTING PROTOCOLS

12.5.1 Destination-Sequenced Distance Vector (DSDV) Protocol

The Destination-Sequenced Distance Vector (DSDV) protocol is a table-driven routing protocol based on the improved version of classical Bellman-Ford routing algorithm. DSDV is based on the Routing Information Protocol (RIP), explained earlier. With RIP, a node holds a routing table containing all the possible destinations within the network and the number of hops to each destination. DSDV is also based on distance vector routing and thus uses bidirectional links. A limitation of DSDV is that it provides only one route for a source/destination pair.

12.5.1.1 Routing Tables

The structure of the routing table for this protocol is simple. Each table entry has a sequence number that is incremented whenever a node sends an updated message. Routing tables are regularly updated when the network topology changes and are propagated throughout the network to keep this information consistent throughout the network.

Each DSDV node maintains two routing tables: one for forwarding packets and the other for advertising incremental routing packets. The routing information sent periodically by a node contains a new sequence number, the destination address, the number of hops to the destination node, and the sequence number of the destination.

When the network topology changes, the node that identifies this change sends an *update* packet to its neighboring nodes. On receipt of an update packet from a neighboring node, a node extracts the information from the packet and updates its routing table using the following algorithm:

12.5.1.2 DSDV Packet Process Algorithm

1. If the new address has a higher sequence number, the node chooses the route with the higher sequence number and discards the old sequence number.
2. If the incoming sequence number is identical to the one belonging to the existing route, a route with the least cost is chosen.
3. All the metrics chosen from the new routing information are incremented.
4. This process continues until all the nodes are updated. If there are duplicate updated packets, the node considers keeping the one with the least-cost metric and discards the rest.

In case of a broken link, a cost of ∞ metric with the next number in sequence is assigned to it to ensure that the sequence number of that metric is always greater than or equal to the sequence number of that node.

The following figure shows a routing table for node 2, whose neighbors are nodes 1, 3, 4, and 8. The dashed lines indicate no communications between any corresponding pair of nodes. Therefore, node 2 has no information about node 8.

Routing Table for Node 2

Destination	Next Hop	Metric	Dest. Seq. No.
1	1	1	123
2	0	0	516
3	3	1	212
4	4	1	168
5	4	2	372
8	1	INF	432

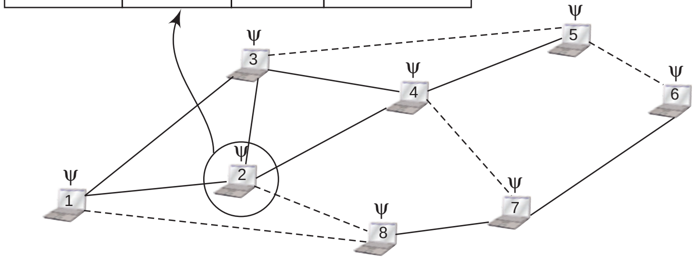


Figure 12.2: A DSDV routing table

The packet overhead of the DSDV protocol increases the total number of nodes in the ad-hoc network. This fact makes DSDV suitable for small networks.

In large ad-hoc networks, the ***mobility rate*** and therefore the ***overhead*** increase, making the network unstable to the point that updated packets might not reach nodes on time.

12.5.2 Cluster-Head Gateway Switch Routing Protocol

The Cluster-Head Gateway Switch Routing (CGSR) protocol is a *table-driven* routing protocol. In a clustering system, a predefined set of nodes are formed into a cluster, controlled by a cluster head that is assigned using a distributed clustering algorithm.

However, with the clustering scheme, a cluster head can be replaced frequently by another node, for several reasons, such as lower-level energy left in the node or a node moves out of contact.

With this protocol, each node maintains two tables: a ***cluster-member table*** and a ***routing table***. The **cluster-member table** records the cluster head for each destination node, and the **routing table** contains the next hop to reach the destination. As with the DSDV protocol, each node updates its cluster-member table on receiving a new update from its neighbors.

12.5.2.1 Clustering and Routing Algorithms

CGSR routing involves cluster routing, whereby a node is required to find the best route over cluster heads from the cluster-member table. The figure below shows an example of routing in an area in which six clusters have been formed.

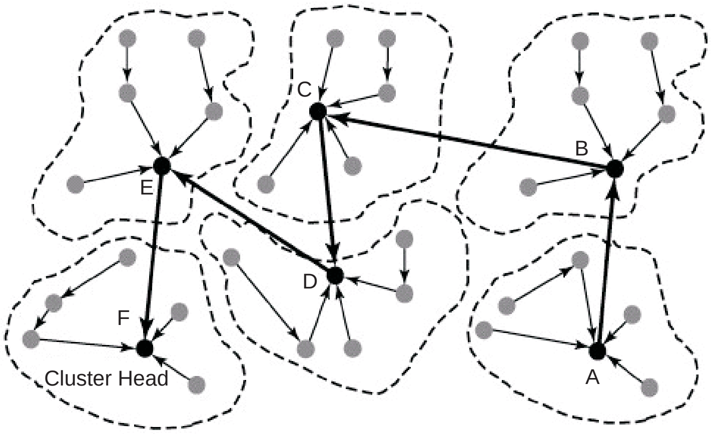


Figure 12.3: Communication with cluster-head gateway switch routing (CGSR) protocol

Here, a node in cluster A is transmitting a packet to a node in cluster F. Nodes within each cluster route their packets to their own associated clusters. The transmitting node then sends its packet to the next hop, according to the routing table entry associated with that cluster head. The cluster head transmits the packet to another cluster head until the cluster head of the destination node is reached. The routing is made through a series of available cluster heads from A to F. Packets are then transmitted to the destination.

12.5.3 Wireless Routing Protocol (WRP)

The Wireless Routing Protocol (WRP) is a table-based protocol maintaining routing information among all nodes in the network. This protocol is based on the distributed Bellman-Ford algorithm. The main advantage of WRP is that it reduces the number of routing loops.

With this protocol, each node in a network maintains four tables, as follows:

1. **Distance table**, which holds the destination, next hop, distance, and predecessors of each destination and each neighbor.
2. **Routing table**, which saves the destination address, next hop, distance, predecessor, and a marker for each destination, specifying whether that entry corresponds to a simple path.
3. **Link-cost table**, which provides the link cost to each neighbor and also the number of periodic update periods elapsed since the node received any error-free message from it.

4. **Message transmission-list table**, which records which updates in an update message are to be retransmitted and which neighbors need to acknowledge the retransmission. The table provides the sequence number of the update message, a retransmission counter, acknowledgements, and a list of updates sent in the update message.

Nodes should either send a message including the update message or a HELLO message to their neighbors. If a node has no message to send, it should send a HELLO message to ensure connectivity. If the sending node is new, it is added to the node's routing table, and the current node sends the new node a copy of its routing table content.

Once it detects a change in a route, a node sends the update message to its neighbors. The neighboring nodes then change their distance entries and look for new possible paths through other nodes. This protocol avoids the *count-to-infinity* issue present in most ad-hoc network protocols. This issue is resolved by making each node perform consistency checks of predecessor information reported by all its neighbors in order to remove looping and make a faster route convergence in the presence of any link or node failure.

12.6 SOURCE INITIATED PROTOCOLS

12.6.1 Dynamic Source Routing (DSR) Protocol

The Dynamic Source Routing (DSR) protocol is an on-demand, or source-initiated, routing protocol in which a source node finds an unexpired route to a destination to send the packet. DSR quickly adapts to topological changes and is typically used for networks in which mobile nodes move with moderate speed.

Overhead is significantly reduced with this protocol, since nodes do not exchange routing table information when there are no changes in the topology. DSR creates multiple paths from a source to a destination, eliminating route discovery when the topology changes.

Similar to most ad-hoc networks, DSR has two phases: **route discovery** and **route maintenance**.

12.6.1.1 Route Discovery and Maintenance

Route discovery begins when a node wants to send packets to another node and no unexpired route to the destination is in its routing table. In such circumstances, the node first broadcasts a *route-request* packet, including the destination address, source address, and a unique identification number. When it receives a route-request packet, the neighboring node it looks at its table; if any route to the requested destination address is already present in the node's route record, the packet is discarded to avoid the looping issue. Otherwise, the node adds its own address to the pre-assigned field of the route-request packet and forwards it to its neighboring nodes.

When the route-request packet reaches a destination node or another intermediate node that has an unexpired route to the destination, this node generates a *route-reply* packet, which contains a route record of the sequence of nodes taken from the source to this node. Once the source receives all the route-reply packets, it updates its routing table with the best path to the destination and sends its packets through that selected route.

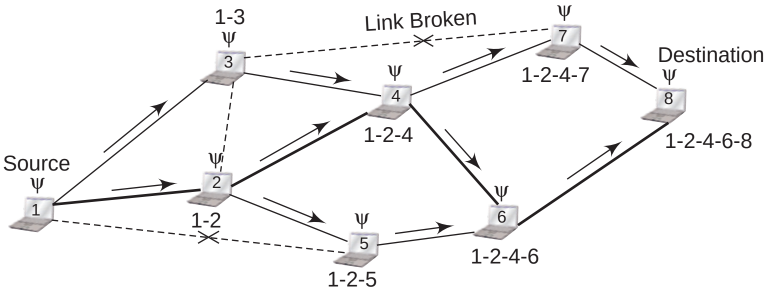


Figure 12.4: Broken link in an ad-hoc network

The figure above shows an ad-hoc network with eight mobile nodes and a broken link (3-7). Node 1 wants to send a message to the destination, node 8. Node 1 looks at its routing table, finds an expired route to node 8, and then propagates route-request packets to nodes 3 and 2. Node 3 finds no route to the destination and so appends the route record 1-3 to the route-request packet and forwards it to node 4.

On receiving this packet, node 7 finds a route to the destination and so stops propagating any route-request packet and instead sends a route-reply packet to the source. The same happens when a route-request packet reaches the destination node 8 with a route record 1-2-4-6. When the source, node 1, compares all the route-reply packets, it concludes that the best route is 1-2-4-6-8 and establishes this path.

Route maintenance in this protocol is fast and simple. In case of a fatal error in the data-link layer, a route-error packet is generated from a failing node. When the route-error packet is received, the failing node is removed from its route cache, and all routes containing that node are truncated. Another route maintenance signal is the acknowledgement packets sent to verify the correct operation of the route links.

12.6.2 Temporally Ordered Routing Algorithm (TORA)

The Temporally Ordered Routing Algorithm (TORA) is a source-initiated routing algorithm and, thus, creates multiple routes for any source/destination pair. The advantage of multiple routes to a destination is that route discovery is not required for every alteration in the network topology. This feature conserves bandwidth usage and increases adaptability to topological changes by reducing communication overhead.

TORA is based on the following three rules:

1. Route creation/discovery
2. Route maintenance
3. Route erasure

TORA uses three types of packets: query packets for route creation, update packets for both route creation and maintenance, and clear packets for route erasure. With TORA, nodes have to maintain routing information about adjacent nodes. This loop-free protocol is distributed based on the concept of link reversal.

Every node maintains a table describing the distance and status of all connected links. When a node has no route to a desired destination a route-creation process starts. A query packet contains the destination ID, and an update packet contains the destination ID and the distance of the node. A receiving node processes a query packet as follows:

- If the receiving node realizes that there are no further downstream links, the query packet is again broadcast. Otherwise, the node discards the query packet.
- If the receiving node realizes that there is at least one downstream link, the node updates its routing table to the new value and broadcasts an update packet.

Once it receives the update packet, a node sets its distance greater than the neighbor from which it received the packet and then rebroadcasts it. The update is eventually received by the source. When it realizes that there is no valid route to the destination, the node adjusts its distance and generates an update packet. TORA performs efficient routing in large networks and in mildly congested networks.

12.6.3 Associative-Based Routing (ABR) Protocol

Associative-Based Routing (ABR) is an efficient on-demand, or source-initiated, routing protocol. ABR's is better than WRPs' network-change adaptation feature, to the extent that it is almost free of loops and is free of packet duplications.

In ABR, the destination node decides the best route, using node associativity. ABR is ideal for small networks, as it provides fast route discovery and creates shortest paths through associativity. Here, the movements of nodes are observed by other nodes in the network.

Each node keeps track of associativity information by sending messages periodically, identifying itself and updating the associativity ticks for its neighbors. If the associativity ticks exceed a maximum, a node has associativity stability with its neighbors. In other words, a low number of associativity ticks show the node's high mobility, and high associativity indicates a node's sleep mode. The associativity ticks can be reset when a node or its neighbor moves to a new location.

Each point-to-point routing in ABR is connection oriented, with all nodes participating in setting up and computing a route. In a point-to-point route, the source node or any intermediate node decides the details of routes. If the communication must be of broadcast type, the source broadcasts the packet in a connectionless routing fashion.

Route discovery is implemented when a node wants to communicate with a destination with no valid route. Route discovery starts with sending a query packet and an await-reply. The broadcast query packet has the source ID, destination ID, all intermediate nodes' IDs, sequence number, CRC, LIVE field and a TYPE field to identify the type of message.

When it receives a query packet, an intermediate node looks at its routing table to see whether it is the intended destination node; otherwise, it appends its ID to the list of all intermediate IDs and rebroadcasts the packet. When it receives all copies of the query packet, the destination node chooses the best route to the source and then sends a reply packet including the best route. This way, all nodes become aware of the best route and thereby make other routes to the destination invalid.

ABR is also able to perform route reconstruction. This function is needed for partial route discovery or invalid route erasure.

12.6.4 Ad-Hoc On-Demand Distance Vector (AODV) Protocol

The Ad-Hoc On-Demand Distance Vector (AODV) routing protocol is an improvement over DSDV and is a source-initiated routing scheme capable of both unicast and multicast routing. AODV establishes a required route only when it is needed as opposed to maintaining a complete list of routes, with DSDV.

AODV uses an improved version of the distance vector algorithm, explained earlier, to provide on-demand routing. AODV offers quick convergence when a network topology changes because of any node movement or link breakage. In such cases, AODV notifies all nodes so that they can invalidate the routes using the lost node or link.

This protocol adapts quickly to dynamic link conditions and offers low processing, memory overhead, and network utilization. Loop-free AODV is self-starting and handles large numbers of mobile nodes. It allows mobile nodes to respond to link breakages and changes in network topology in a timely manner. The algorithm's primary features are as follows:

- It broadcasts packets only when required.
- It distinguishes between local connectivity management and general maintenance.
- It disseminates information about changes in local connectivity to neighboring mobile nodes that need this information.
- Nodes that are not part of active paths neither maintain any routing information nor participate in any periodic routing table exchanges.
- A node does not have to find and maintain a route to another node until the two nodes communicate. Routes are maintained on all nodes on an active path. For example, all transmitting nodes maintain the route to the destination.

AODV can also form multicast trees that connect multicast group members. The trees are composed of group members and the nodes needed to connect them and is similar to multicast protocols discussed earlier.

12.6.4.1 Routing Process

A route is active as long as data packets periodically travel from a source to a destination along that path. In other words, an active route from a source to a destination has a valid entry in a routing table. Packets can be forwarded only on active routes. Each mobile node maintains a routing table entry for a potential destination. A routing table entry contains

- Active neighbors for a requested route
- Next-hop address
- Destination address
- Number of hops to destination
- Sequence number for the destination
- Expiration time for the route table entry (timeout)

Each routing table entry maintains the addresses of the active neighbors so that all active source nodes can be notified when a link along the route to the destination breaks. For each valid route, the node also stores a list of precursors that may transmit packets on this route. These precursors receive notifications from the node when it detects the loss of the next-hop link.

Any route in the routing table is tagged with the destination's sequence number; an increasing number set by a counter, and managed by each originating node, to ensure the freshness of the reverse route to a source.

The sequence number is incremented whenever the source issues a new route-request message. Each node also records information about its neighbors with bidirectional connectivity. The insertion of a sequence number guarantees that no routing loops form even when packets are delivered out of order and under high node mobility. If a new route becomes available to the requested destination, the node compares the destination sequence number of the new incoming route with the one stored in the routing table for the current route.

The existing entry is updated by replacing the current entry with the incoming one if any of the following conditions exist:

- The current sequence number in the routing table is marked invalid.
- The new incoming sequence number is greater than the one stored in the table.
- The sequence numbers are the same, and the new hop count is smaller than the one for the current entry.

Once the source stops sending data packets on an established connection, the routes time out and are eventually deleted from the intermediate-node routing tables. At the reverse-path entry of any routing table, a request-expiration timer purges the reverse-path routing entries from the nodes that do not lie on the route from the source to the destination.

When an entry is used to transmit a packet, the timeout for the entry is reset to the current time plus the active-route timeout. The routing table also stores the routing caching time out, which is the time after which the route is considered to be invalid.

12.6.4.2 Route Discovery and Establishment

Suppose that a source node does not have information about a destination node, perhaps because it is unknown to the source node or a previously valid path to the destination expires or is marked invalid. In such cases, the source node initiates a route-discovery process to locate the destination.

Route discovery is done by broadcasting a route request (RREQ) packet to neighbors, which in turn is forwarded to their neighbors until the destination node is reached. If it receives an already processed RREQ, a node discards the RREQ and does not forward it.

Neighboring nodes update their information and set up backward pointers to the source node in their route tables. Each neighbor either responds to the RREQ by sending back a route-reply RREP to the source or increases the hop count and broadcasts the RREQ to its own neighbors; in this case, the process continues.

An RREQ packet contains the following information:

- Source address
- A unique RREQ
- Destination address
- Source sequence number
- Destination sequence number
- Hop count
- Lifetime

When an RREQ packet arrives at an intermediate node on a route, the node first updates the route to the previous hop. The node then checks whether the available route is current and completes this check by comparing the destination sequence number in its own route entry to the destination sequence number in the RREQ packet. If the destination sequence number is greater than the one available in the intermediate node's routing table, the intermediate node must not use its recorded route to respond to the RREQ. In this case, the intermediate node rebroadcasts the RREQ to its neighboring node.

On receipt of an RREQ, an intermediate node maintains the address of each neighbor from which the first copy of the packet is received in their route tables and establishes a reverse path. Therefore, if additional copies of the same RREQ are received, they are discarded. When the RREQ reaches the destination node, it responds by sending a route reply (RREP) packet back to the neighbor from which the RREQ was first received. As the RREP is routed back, nodes along this reverse path setup forward route entries in their routing tables, which indicates the node from which the RREP came.

Any intermediate node checks whether it has received an RREQ with the same source node IP address and RREQ within at least the last path-discovery time. If such an RREQ has been received, the node drops the newly received RREQ. The reverse-route entries are maintained long enough for the RREQ packet to pass through the network and produce a reply to the sender. For the RREQ that is not dropped, the node increments the hop count and then searches for a reverse route to the source.

At this point, a routing timer associated with each route times out and deletes the entry if it has not received any RREP or is not used within a specified time. The protocol uses destination sequence-numbers to ensure that links are free of loops at all times and thus avoids counting to infinity. The destination address field is incremented every time a source issues a new RREQ.

The following figure shows the process of signals with AODV from node 1 to node 8. To establish a connection, source node 1 searches in its table for a valid route to destination node 8.

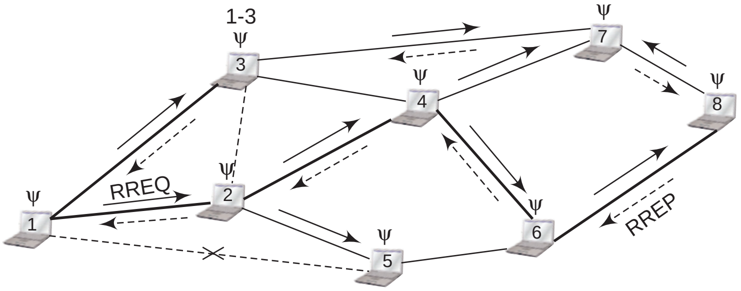


Figure 12.5(a): Working of AODV for routing

In the figure, an RREQ reaches the destination for the first time through path 1-2-4-6-8. The destination then issues an RREP packet to the source. After a while, the destination receives another RREQ, this time through path 1-3-7-8. The destination evaluates this path, and finds that path 1-3-7-8 is better, and then issues a new RREP packet, telling the source to discard the other reply.

This means that, while transmitting, if it receives a better RREP packet containing a greater sequence number or the same sequence number with a smaller hop count, the source may update its routing information for that destination and switch over to the better path. As a result, a node ignores all less-desired RREPs received while transmitting.

At the reverse-path entry of any routing table, a request-expiration timer purges the reverse-path routing entries from the nodes that do not lie on the route from the source to the destination. When an entry is used to transmit a packet, the timeout for the entry is reset to the current time plus the active-route timeout.

The routing table also stores the routing caching timeout, which is the time after which the route is considered invalid. The following figure depicts the situation where a timeout has occurred. From the source node 1 to the destination node 8, two routes 1-2-4-6-8 and 1-3-7-8, are found. However, the RREP at the intermediate node 7 exceeds the time allowed to be released. In this case, route 1-3-7-8 is purged from the involving routing tables.

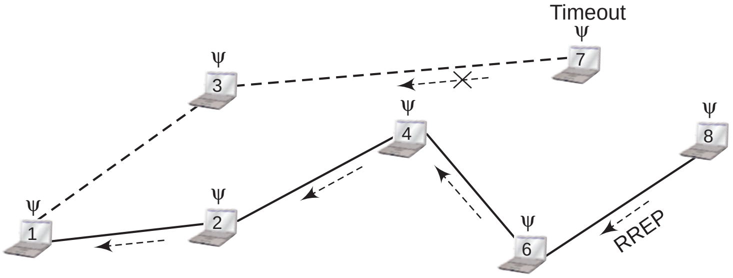


Figure 12.5(b): Timeout in AODV routing

12.6.4.3 Route Maintenance

After it knows how to establish a path, a network must maintain it. In general, each forwarding node should keep track of its continued connectivity to its active next hops. If a source node moves, it can reinitiate route discovery to find a fresh route to the destination.

When a node along the route moves, its upstream neighbor identifies the move and propagates a link-failure notification message to each of its upstream neighbors. These nodes forward the link-failure notification to their neighbors, and so on, until the source node is reached. The source node may reinitiate path-discovery if a route is still desired.

When the local connectivity of a mobile node is required, each mobile node can get information about other nodes in its neighborhood by using local broadcasts known as HELLO messages. A node should use HELLO messages only if it is part of an active route. A node that does not receive a HELLO message from its neighbors along an active route sends a link-failure notification message to its upstream node on that route.

When it moves during an active session, a source node can start the route-discovery process again to find a new route to the destination node. If the destination or the intermediate node moves, a special RREP is sent to the affected source nodes.

Periodic HELLO messages can normally be used to detect link failures. If a link failure occurs while the route is active, the upstream node of the breakage propagates a route-error (RERR) message. An RERR message can be either broadcast or unicast.

For each node, if a link to the next hop is undetectable, the node should assume that the link is lost and take the following steps.

1. Make all related existing routes invalid.
2. List all destination nodes that would potentially be affected.
3. Identify all neighboring nodes that may be affected.
4. Send an RERR message to all such neighbors.

As discussed earlier, some next-hop nodes in a network might be unreachable. In such cases, the upstream node of the unreachable node propagates an unsolicited RREP with a fresh sequence number, and the hop count is set to infinity to all active upstream neighbors. Other nodes listen and pass on this message to their active neighbors until all nodes are notified. AODV finally terminates the unreachable node (broken associated links).

12.7 JOINING A NEW NODE TO AN AD-HOC NETWORK

A new node can join an ad-hoc network by transmitting a HELLO message containing its identity and sequence number. When a node receives a HELLO message from a neighbor, the node makes sure that it has an active route to it or, if necessary, creates one. After this update, the current node can begin using this route to forward data packets.

In general, nodes in a network may learn about their neighbors when a node receives a normal broadcast message or HELLO message. If the HELLO message is not received from the next hop along an active path, the active neighbors using that next hop are sent notification of a link break.

A node receiving HELLO messages should be part of an active route in order to use them. In every predetermined interval, active nodes in a network check whether they received HELLO messages from their neighbors.

If it does not receive any packets within the hello interval, a node broadcasts a HELLO message to all its neighbors. Neighbors that receive this packet update their local connectivity information. Otherwise, if it does not receive any HELLO messages for more than some predetermined time, a node should assume that the link to this neighbor failed.

12.8 SECURITY OF AD-HOC NETWORKS

Because of dynamic topological changes, ad-hoc networks are vulnerable at the physical link, as they can easily be manipulated. An intruder can easily attack ad-hoc networks by loading available network resources, such as wireless links and energy (battery) levels of other users, and then disturb all users.

Attackers can also disturb the normal operation of routing protocols by modifying packets. The intruder may insert spurious information into routing packets, causing erroneous routing table updates and thus misrouting.

Some other security issues of ad-hoc networks are listed below.

- **Limited computational capabilities**. Typically, nodes in ad-hoc networks are modular, independent, and limited in computational capability and therefore may become a source of vulnerability when they handle public-key cryptography during normal operation.
- **Limited power supply**. Since nodes normally use battery as power supply, an intruder can exhaust batteries by creating additional transmissions or excessive computations to be carried out by nodes.
- **Challenging key management**. Dynamic topology and movement of nodes in an ad-hoc network make key management difficult if cryptography is used in the routing protocol.

In any network, routing information can give an attacker access to relationships among nodes and their IP addresses. Especially in ad-hoc networks, an attacker may be able to bring the network down.

12.9 TYPES OF ATTACKS

Attacks in ad-hoc networks are either passive or active. In a passive attack, the normal operation of a routing protocol is not interrupted. Instead, an intruder tries to gather information by listening. Active attacks can sometimes be detectable and thus are less important. In an active attack, an attacker can insert some arbitrary packets of information into the network to disable it or to attract packets destined to other nodes.

12.9.1 Pin Attack

With the pin, or black-hole, attack, a malicious node pretends to have the shortest path to the destination of a packet. Normally, the intruder listens to a path set-up phase and, when learns of a request for a route, sends a reply advertising a shortest route.

Then, the intruder can be an official part of the network if the requesting node receives its malicious reply before the reply from a good node, and a forged route is set up. Once it becomes part of the network, the intruder can do anything within the network, such as undertaking a denial-of-service attack.

12.9.2 Location-Disclosure Attack

By learning the locations of intermediate nodes, an intruder can find out the location of a target node. The location-disclosure attack is made by an intruder to obtain information about the physical location of nodes in the network or the topology of the network.

12.9.3 Routing Table Overflow

Sometimes, an intruder can create routes whose destinations do not exist. This type of attack, known as the routing table overflow overwhelms the usual flow of traffic, as it creates too many dummy active routes.

This attack has a profound impact on proactive routing protocols, which discover routing information before it is needed, but minimal impact on reactive routing protocols, which create a route only when needed.

12.9.4 Energy-Exhaustion Attack

Battery-powered nodes can conserve their power by transmitting only when needed. But an intruder may try to forward unwanted packets or request repeatedly fake or unwanted destinations to use up the resources at the node.

12.10 CRITERIA FOR A SECURE ROUTING PROTOCOL

In order to prevent ad-hoc networks from attacks and vulnerability, a routing protocol must possess the following properties:

- **Authenticity**. When a routing table is updated, it must check whether the updates were provided by authenticated nodes and users. The most challenging issue in ad-hoc networks is the lack of a centralized authority to issue and validate certificates of authenticity.

- **Integrity of information**. When a routing table is updated, the information carried to the routing updates must be checked for eligibility. A misleading update may alter the flow of packets in the network.
- **In-order updates**. Ad-hoc routing protocols must contain unique sequence numbers to maintain updates in order. Out-of-order updates may result in the propagation of wrong information.
- **Maximum update time**. Updates in routing tables must be done in the shortest possible time to ensure the credibility of the update information. A timestamp or time-out mechanism can normally be a solution.
- **Authorization**. A non-forgeable credential along with the certificate authority issued to a node can determine all the privileges that the node can have.
- **Routing encryption**. Encrypting packets can prevent unauthorized nodes from reading them, and only those routers having the decryption key can access messages.
- **Route discovery**. It should always be possible to find any existing route between two points in a network.
- **Protocol immunization**. A routing protocol should be immune to intruding nodes and be able identify them.
- **Node-privacy location**. The routing protocol must protect the network from spreading the location or other non-public information of individual nodes.
- **Self-stabilization**. If the self-stabilization property of ad-hoc network performs efficiently, it must stabilize the network in the presence of damages continually received from malicious nodes.
- **Low computational load**. Typically, an ad hoc node is limited in powers as it uses a battery. As a result, a node should be given the minimal computational load to maintain enough power to avoid any denial-of-service attacks from low available power.

SUMMARY

A wireless ad-hoc network supports “independent” wireless and mobile communication systems. A mobile user in fact acts as a routing node. Routing protocols can be centralized versus distributed, static versus adaptive, and table driven versus source initiated routing.

Table-driven routing protocols find routes to all possible destinations before they are needed. The routes are recorded in nodes’ routing tables and are updated within predefined intervals. Examples of such protocols are DSDV, CGSR, and WRP. CGSR had a better converging capability than others do. Source-initiated routing protocols, such as DSR, ABR, TORA, and AODV, create routes only when they are requested by source nodes. AODV is very popular owing to its ability to stabilize the routing and its better security. Security is a critical issue in ad-hoc networks. Security vulnerability to various types of attacks can be minimized by meeting specific security criteria.

13

Sensor Networks and Related Advanced Technologies

Introduction

Chemical, biological, or solar sensors can be networked together as a sensor network to strengthen the power of sensing. A sensor network is controlled through a software core engine. The network is typically un-wired, or wireless, but may also be wired. Sensor networks are designed to be self-configuring such that they can gather information about a large geographical area or about movements of an object for surveillance purposes.

Sensor networks can be used for target tracking, environmental monitoring, system control, and chemical or biological detection. In Military applications, sensor networks can enable soldiers to see around corners and to detect chemical and biological weapons long before they get close enough to cause harm. Civilian uses include environmental monitoring, traffic control, and providing health care monitoring for the elderly while allowing them more freedom to move about.

13.1 CLUSTERING IN SENSOR NETWORKS

The region being sensed is normally partitioned into equally loaded clusters of sensor nodes, as shown below.

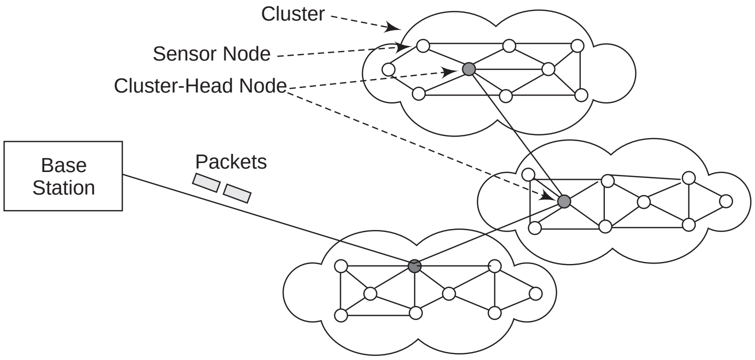


Figure 13.1: Cluster of sensor nodes

A cluster in a sensor network resembles a domain in a computer network. In other words, nodes are inserted in the vicinity of a certain predefined region, forming a cluster. Different types of sensors can also be deployed in a region. Thus, a sensor network is typically cluster based and has irregular topology.

The most effective routing scheme in sensor networks is normally based on the energy (battery level) of nodes. In such routing schemes, the best path has the highest amount of total energy. The network of such sensing nodes is constructed with identical sensor nodes, regardless of the size of the network.

Here, in the figure we see three clusters are interconnected to the main base station, each cluster contains a cluster head responsible for routing data from its corresponding cluster to a base station.

Communicating nodes are normally linked by a wireless medium, such as radio. The wireless sensor node is equipped with a limited power source, such as a battery or even a solar cell, where there is enough sunlight exposure on the node.

However, a solar cell may not be the best choice as a power supply, owing to its weight, volume, and expense. In some application scenarios, sensor-node lifetime depends on the battery lifetime.

Removal of dead nodes can cause significant topological changes and may require packet rerouting. As a result, power management is a key issue in system design, node design, and communication protocol development. In short, efficient energy-conscious clustering and routing algorithms can potentially prolong the network lifetime.

13.1.1 Protocol Stack

The algorithms developed for wireless ad-hoc networks cannot be used for sensor networks, for several reasons. One is that the number of sensor nodes is typically much more than in a typical ad-hoc network, and sensor nodes, unlike ad-hoc nodes, are prone to permanent failure.

In addition, sensor nodes normally use broadcast rather than point-to-point communication with its limited power and memory. Unlike computer networks, sensor nodes do not have global ID, since a typical packet overhead can be too large for them.

Following is the protocol architecture for sensor networks. The protocol stack combines power efficiency and least-cost-path routing. This protocol architecture integrates networking protocols and power through the wireless medium and promotes cooperative efforts of sensor nodes.

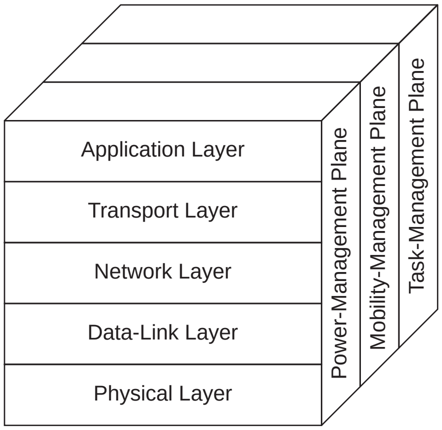


Figure 13.2: Protocol architecture for sensor networks

The protocol stack consists of the physical layer, data-link layer, network layer, transport layer, and application layer, backed by a power-management plane, mobility-management plane, and task-management plane.

The physical layer is responsible for robust modulation, transmission, and receiving signals. Media access control (MAC) at the data-link layer must minimize packet collision with neighboring nodes, as power is a restricted factor.

The network layer routes packets provided by the transport layer. The application layer uses software for preparation of data on an event. The power-management plane monitors the sensor's power level among the sensor nodes and manages the amount of power a sensor node has used.

Most of the sensor network routing techniques and sensing tasks require an accurate knowledge of location. Thus, a sensor node commonly has a location-finding system. A mobilizer may sometimes be needed to move sensor nodes to carry out assigned tasks. Sensor network routing protocols must be capable of self-organizing.

For these purposes, a series of energy-aware MAC, routing, and clustering protocols have been developed for wireless sensor networks. Most of the energy-aware MAC protocols aim to either adjust the transmission power or keep transceivers off as long as possible.

13.1.2 Sensor Node Structure

A sensor node, as shown below, consists mainly of a sensing unit, a processing unit and memory, a self-power unit, and a wireless transceiver component, as well as a self- and remote-testing unit, a synchronizing and timing unit, a routing table, and security units. Since nodes in a network are not physically accessible once they are deployed in the field, they are not worth being brought under test. An option is an on-board remote self-testing unit for the node on a routine basis.

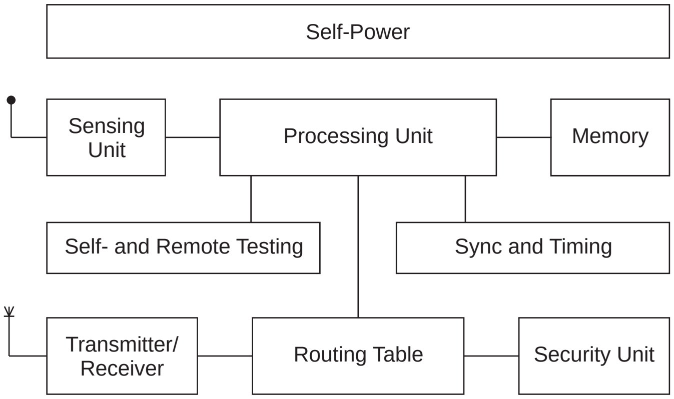


Figure 13.3: Structure of a sensor node

Each node must determine its location. This task is carried out by a location-finding system based on the global positioning system (GPS). All the processes within the sensor node are synchronized by a local clocking and synchronizing system.

The *communication* and *security protocol* units are in fact part of the processing unit. These two units are responsible for computing the best path for networking and security of the data being transmitted. The three main blocks of the sensor node; that is, **sensing unit**, **processing and memory unit,** and **power unit** are described in more detail in the following subsections.

13.1.2.1 Sensing Unit

The sensing unit consists of a sensor and an analog-to-digital converter. A smart sensor node consists of a combination of multiple sensors. The analog signals produced by the sensors, based on the observed event, are converted to digital signals by the converter and then fed into the processing unit. The sensing unit collects data externally and interacts with the central processor at the heart of the node.

13.1.2.2 Processing and Memory Unit

The processing unit performs certain computations on the data and, depending on how it is programmed, may send the resulting information out to the network. The processing unit, which is generally associated with memory, manages the procedures that make the sensor node collaborate with the other nodes to carry out the assigned sensing task.

The central processor determines what data needs to be analyzed, stored, or compared with the data stored in memory. The streamed data from the sensor input is processed as it arrives. The database in memory stores an indexed data list to be used as a reference to detect an event.

Since sensing nodes are typically tiny and many nodes are engaged in a network, the communication structure makes use of a hierarchically arranged self-routing network through cluster heads.

In smart wireless sensor networks, a tiny processor and a tiny database are used in a node. Thousands of such nodes are spread on fields to power up the sensing task, as in the deployment of numerous small intelligent sensor nodes in a battlefield to monitor enemy movements.

By inserting self-organizing capability into a sensor network, a smart node can extract data, compare it with the data stored in its memory database, and process it for relevance before relaying it to its central base station.

13.1.2.3 Self-Power Unit

A sensor node is supposed to be mounted in a small physical unit, limiting space for the battery. Moreover, the random distribution of sensors makes it impossible to periodically recharge or exchange batteries.

In most types of sensor networks, the power unit in a sensor node is the most important unit of the node because the liveliness and existence of a node depend on the energy left in the node, and the routing in the sensor network is based on the algorithm that finds a path with the most energy. Thus, it is essential to use energy-efficient algorithms to prolong the life of sensor networks.

The main task of the sensor node is to identify events, to process data, and then to transmit the data. The power of a node is consumed mainly in the transmitter and receiver unit. The sensor node can be supplied by a self-power unit, battery, or solar cells.

13.2 COMMUNICATION ENERGY MODEL

IEEE standards as 802.11a, b, and g provide a wide range of data rates: 54, 48, 36, 24, 18, 12, 9, and 6 Mb/s. This range reflects the trade-off between the transmission range and data rate intrinsic in a wireless communication channel. An accurate energy model is crucial for the development of energy-efficient clustering and routing protocols. The energy consumption, E, for all components of the transceiver in watts is summarized as:

$$E = \Theta + \eta\omega d^n,$$

where θ is the distance-independent term that accounts for the overhead of the radio electronics and digital processing, and $\eta\boldsymbol{\omega}\mathbf{d}^{\mathbf{n}}$, is the distance-dependent term, in which η represents the amplifier inefficiency factor, ω is the free-space path loss, **d** is the distance, and **n** is the environmental factor.

Based on an environmental condition, n can be a number between 2 and 4, and η specifies the inefficiency of the transmitter when generating maximum power ωd^n at the antenna. Clearly, the distance-dependent portion of total energy consumption depends on the real-world transceiver parameters, θ, η, and the path attenuation ωd^n. If the value of θ overshadows $\eta\omega d^n$, the reduction in the transmission distance through the use of multihop communication is not effective.

In theory, the maximum efficiency of a power amplifier is 48.4 percent. However, practical implementations show that the power-amplifier efficiency is less than 40 percent. Therefore, θ is calculated assuming that η = 1/0.4 = 2.5. Using the above equation, we can express the energy consumption of a transmitter and a receiver, E_T and E_R, respectively, by

$$E_T = \theta_T + \eta\omega d^n$$

and

$$E_R = \theta_R,$$

where θ_T and θ_R are the distance-dependent terms for the transmitter and the receiver, respectively.

Although, sensor nodes in general generate data in low rates, they can transmit the information using wireless high-speed modulation and techniques. The following table lists the Energy Consumption standards for Wireless Sensor Networks:

Table 13.1: Energy consumption standards for wireless sensor networks

IEEE Standard	Max. Output Power, ωd^n (dBm)	Total Power Consumption (W)	θ(W)	$\eta \times \omega d^n$(W)
802.11a	+14	1.85 (E_{TX}) 1.20 (E_{RX})	2.987	0.0625
802.11b	+21	1.75 (E_{TX}) 1.29 (E_{RX})	2.727	0.3125
802.11g	+14	1.82 (E_{TX}) 1.40 (E_{RX})	3.157	0.0625

13.2.1 Multi-Hop Communication Efficiency

Considering the impact of real-world radio parameters and multi-rate communication, we should re-evaluate the effectiveness of multi-hop communications. Since a multi-rate communication reduces energy consumption for shorter distances by switching to higher data rates, multi-hop communication can conserve energy.

The traditional objective of multi-hop communication is to divide the transmission distance into a number of hops, m, and to relatively conserve energy, by means of

$$E = m(\theta + \omega(d/m)^n)$$

However, if the division of transmission distance happens when the maximum range is less than 18.75 m for standard 802.11g, the data rate remains constant, and total energy consumption multiplies by the number of hops.

Since sensor networks deal with two or even three-dimensional spaces multi-hop efficiency depends on the network scale and density.

13.3 CLUSTERING PROTOCOLS

Clustering protocols specify the topology of the hierarchical non-overlapping clusters of sensor nodes. A robust clustering technique is essential for self-organizing sensor networks. An efficient clustering protocol ensures the creation of clusters with almost the same radius and cluster heads that are best positioned in the clusters.

Since every node in a clustered network is connected to a cluster head, route discovery among cluster heads is sufficient to establish a feasible route in the network. For a large sensor network, clustering can simplify multi-hop route discovery and limit the number of transmissions compared to a flat, non-clustered network.

13.3.1 Classification of Clustering Protocols

Clustering techniques can be either centralized or decentralized. Centralized clustering algorithms require each sensor node to send its individual information, such as energy level and geographical position, to the central base station.

Based on a predefined algorithm, a base station calculates the number of clusters, their sizes, and the cluster heads' positions and then provides each node with its newly assigned duty.

Decentralized clustering techniques create clusters without the help of any centralized base station. An energy-efficient and hierarchical clustering algorithm can be such a way whereby each sensor node becomes a cluster head with a probability of p and advertises its candidacy to nodes that are no more than k hops away from the cluster head.

Given the limited transmission range of wireless sensor nodes, a hierarchical structure with an arbitrary number of levels has its limitations. As the number of hierarchical levels grows, the distance between upper-level cluster heads may increase to the point that they are no longer able to communicate with one another.

The Low-Energy Adaptive Clustering Hierarchy (LEACH) algorithm and the Decentralized Energy-Efficient Cluster Propagation (DEEP) protocol are two examples of the decentralized clustering protocols.

13.3.2 LEACH Clustering Protocol

The Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol is a decentralized clustering algorithm that does not offer a complete energy-optimization solution, as it has no strategy for specifying cluster-head positioning and distribution. LEACH is an application-specific protocol architecture that aims to prolong network lifetime by periodic re-clustering and change of the network topology.

LEACH is divided into rounds consisting of a clustering phase and a steady-state phase for data collection. At the start of each round, a sensor node randomly chooses a number between 0 and 1 and then compares this number to a calculated threshold called T(n).

If T(n) is larger than the chosen number, the node becomes a cluster head for the current round. The value T(n) is calculated using the following formula:

$$\mathrm{T}(n) = \begin{cases} \dfrac{p}{1 - p\left(r \bmod \left(1/p\right)\right)} & \text{for } n \in G \\ 0 & \text{otherwise} \end{cases}$$

where p is the ratio of the total number of cluster heads to the total number of nodes, r is the number of rounds, and G is a set of nodes that have not been chosen as cluster heads for the last $1/p$ rounds.

For the first round ($r = 0$), T(n) is equal to p, and nodes have an equal chance to become cluster head. As r gets closer to $1/p$, T(n) increases, and nodes that have not been selected as cluster head in the last $1/p$ rounds have more chance to become cluster head.

After $1/p - 1$ rounds, T(n) is equal to 1, meaning that all the remaining nodes have been selected as cluster head. Thus, after $1/p$ rounds, all the nodes have had a chance to become a cluster head once.

Since being the cluster head puts a substantial burden on the sensor nodes, this ensures that the network has no overloaded node that runs out of energy sooner than the others.

After cluster heads are self-selected, they start to advertise their candidacy to the other sensor nodes. When it receives advertisements from more than one cluster-head candidate, a sensor node starts to make a decision about its corresponding cluster head. Each node listens to the advertisement signals and chooses the candidate whose associated signal is received with higher power.

This ensures that each sensor node chooses the closest candidate as cluster head. The LEACH algorithm is distributed, as it can be accomplished by local computation and communication at each node, rather than the transmission of all the nodes' energy level and geographical position to a centralized point. However, cluster heads are chosen randomly, and there is no optimization in terms of energy consumption.

13.3.3 DEEP Clustering Protocol

The Decentralized Energy-Efficient Cluster Propagation (DEEP) protocol that establishes clusters with uniformly distributed cluster heads. This protocol balances the load among all the cluster heads by keeping the clusters' radii fairly equal. This protocol is completely decentralized, and there is no need for any location-finder device or hardware.

The protocol starts with an initial cluster head, and forms new cluster-head candidates gradually by controlling the relative distance between a pair of cluster heads and the circular radius of each cluster. Owing to the balanced load among cluster heads, periodic reclustering is not necessary, and operational expenses caused by frequent reclustering are therefore eliminated.

An efficient path-selection algorithm for nodes that are placed more than l meters away from a cluster head is necessary in order to find the optimum two-hop or three-hop path. Although direct transmission to a cluster head can eliminate the overhead created by the route set-up packets, its efficiency is questionable, owing to the limited transmission range.

In order to avoid the frequent control signal transmission and extra power consumption associated with that, a cluster head can be placed at the center of the cluster, with sensor nodes positioned closer than l meters around it.

In this case, cluster members can send the data packets directly to the cluster head without the need for any route set-up protocol, while efficiency has already been achieved through the choice of cluster shape and cluster size.

In order to explain the details of this algorithm, control signals and protocol parameters need to be introduced:

- Control signals: (1) cluster-head declaration signal or (2) cluster-head exploration signal
- Membership search signal with control parameters: declaration range (d_r), exploration range (d_{r1}, d_{r2}), minimum number of members (m_n), E_{rc1}, and E_{rc2}.

Protocol-control parameters are application-specific choices and can be defined prior to network deployment. The DEEP protocol forms clusters by starting with an initial cluster head that can be chosen prior to network deployment.

The initial cluster head starts the cluster set-up phase by propagating cluster-head declaration signals within the range of d_r. This means that the cluster-head candidate chooses an appropriate data rate and signal output power so that it can reach nodes that are less than d_r away from the sender.

At this point, sensor nodes that receive the declaration signal accept the corresponding cluster head as a leader. They can then estimate their relative distance to the candidate by looking at the received signal's energy level. Once they know the relative distance to the cluster head, they can conserve energy by adjusting the transmission speed to the appropriate value and switching to sleep mode. Now, the initial cluster-head candidate propagates the cluster-head exploration signal within the range of d_{r2}, as shown.

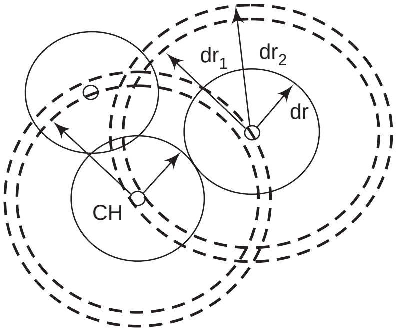


Figure 13.4: Propagates of cluster-head exploration signal

After a new cluster-head candidate is assigned, it sends a declaration signal within the range of d_r to find new cluster members. If two candidates can hear each other's declaration signal, they are too close to each other to be considered cluster-head candidates.

Therefore, one of them is eliminated through a negotiation phase. Whenever it receives a declaration signal, a cluster head informs the sender of the message, using an acknowledgment message. A cluster head that receives the acknowledgment sends a dissolution message and informs all nodes within the range of d_r about its elimination.

A node that receives a declaration signal from more than one candidate chooses the candidate whose associated signal is received with a higher power.

At this point, all confirmed cluster heads propagate exploration signals and search for new cluster-head candidates. Nodes that have already been chosen as cluster head or member ignore the cluster-head exploration or declaration signals.

Therefore, this advertisement process terminates automatically when all the nodes in the field belong to a cluster. At this point, the algorithm might have produced some clusters with a very small number of members. Therefore, a cluster whose total number of members is smaller than the minimum number of members, m_n, is dissolved, and all its members, including its cluster head, initiate a membership-search signal.

After this process is completed, nodes listen to the responses from the local cluster heads and choose the closest cluster head, based on the received signal power. At the end, if the timeout has been reached, a sensor node that has not received any control signal sends a membership-search signal and chooses the closest cluster head as leader. The following algorithm summarizes the core segment of the DEEP protocol.

Begin DEEP Clustering Algorithm

1. Initialize: Initial cluster head finds cluster members by sending "cluster-head declaration."
2. Initial cluster head finds new cluster-head candidates by sending "cluster-head exploration signal."
3. Repeat: Cluster-head candidates that are placed on the (d_{r1}, d_{r2}) ring find cluster members.
4. Nodes that receive more than one cluster-head declaration choose the closest cluster head, based on the received signal energy.
5. Cluster-head candidates that receive a cluster-head declaration signal negotiate with the sender, and one of them gets eliminated.
6. Confirmed cluster heads send "cluster-head exploration" signals to find new cluster-head candidates (Go to step 4).
7. Finalize: If the number of members in a cluster is less than m_n, all the members find new clusters by sending the membership-search signal.
8. At the end, a node that has not received any control signal sends the membership-search signal.

DEEP has several advantages over other clustering protocols. With DEEP, a sensor node can either select itself as a cluster head by receiving a cluster-head exploration signal or join a cluster by receiving a cluster-head declaration signal. After the execution of the protocol, all the sensor nodes are covered and belong to only one cluster.

This clearly shows that this protocol is completely decentralized. In addition, for the execution of DEEP, there is no need for any location-finder hardware, such as the global positioning system (GPS) or a position-estimation protocol that puts extra overhead on sensor nodes.

13.3.4 Reclustering

In order to prevent overutilization of some sensor nodes, clustering technique should ensure that the cluster-head responsibility rotates among all sensor nodes. To achieve this, reclustering is performed periodically in LEACH. However, every round of reclustering requires several control-signal exchanges among self-elected cluster heads and sensor nodes.

The reclustering process in DEEP is based on one small shift in the initial cluster head. When the current period of cluster setting is finished, the current initial CH chooses the nearest node that has never acted as an initial cluster head. This newly chosen initial cluster head starts the clustering process and creates a totally different cluster-head constellation.

13.4 ROUTING PROTOCOLS

After clusters with well-distributed cluster heads have been established in a network, energy-conscious routing is essential in order to set communication routes among cluster heads in a two-level hierarchical system.

Similar to computer networks, routing protocols in sensor networks can be classified as either intra-cluster or inter-cluster. We look at both categories here.

The fundamental concept behind them is much the same as the concept behind intra-domain and inter-domain routings. Assuming that every node in a cluster can act as a relay node, there could be a large number of possible routes from a source to a sink.

Because of the limited transmission range associated with low-power wireless technologies cluster-head packets cannot reach the base station unless other cluster heads act as relay nodes.

Two major approaches can be used for routing and selecting the best path in a sensor network, as follows:

1. Centralized routing, whereby the routing decision is made by a single command center
2. Distributed routing, whereby the routing decision is made in a distributed fashion by several entities

Distributed routing algorithms are further classified as proactive or reactive. With proactive routing algorithms, such as link-state routing and distance-vector routing, nodes keep a routing table that contains next-hop information to every node in the network.

Reactive routing protocols set a route to the desirable destination only when it is needed. Note that none of the ad hoc network protocols explained earlier consider energy consumption.

Another group of on-demand reactive routing protocols address the exclusive issues of wireless sensor network. For example, directed diffusion introduces a concept of "interest" propagation whenever a node wants to send data or a source needs to ask for it.

With this type of protocol, flooding the network with interest signals establishes a path from a sink to every possible source (spanning tree).

13.4.1 Intracluster Routing Protocols

A routing algorithm within a cluster can be either direct or multi-hop. In a direct routing algorithm, the cluster head as the destination for all cluster nodes is located in the center of the cluster, so all nodes can communicate with the cluster head directly, as shown.

Note that in this figure, two nodes cannot reach the destination, as they are located far from it. The number shown in each node indicates the level of energy the corresponding node has.

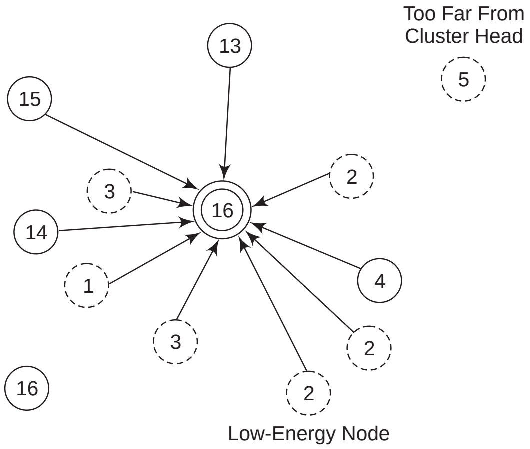


Figure 13.5: Direct routing in a cluster. The number associated with each node indicates a normalized value of the remaining energy in that node.

In a multihop routing algorithm, a node can face multiple hops in order to reach the destination. If a multihop algorithm is used for the centralized clustering procedure, the algorithm aims to choose the appropriate next neighbor for each node, using a central command node. Typically, a central command node collects the information about direct paths' costs and geographical positions of the nodes and finds the best path.

The following figure shows a routing implementation. Sensor nodes are usually scattered in the field. A packet from a node is routed to a neighboring node that exhibits the highest amount of energy. The energy is an indication of the node's battery level. The number associated with each node indicates a normalized value of the remaining energy in that node.

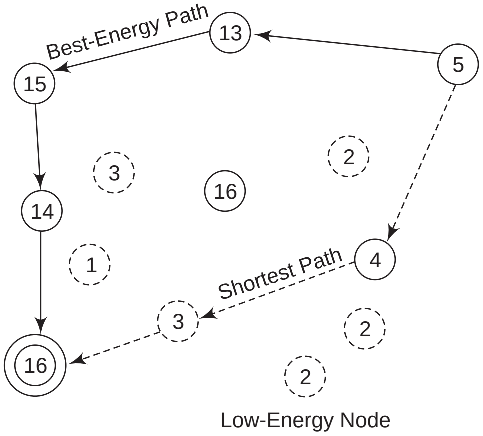


Figure 13.6: Multihop routing in a cluster in which the number associated with each node indicates a normalized value of the remaining energy in that node

In the figure, we see two paths from a node to a cluster-head node. One path involves the shortest distance in terms of hop counts; the other one uses the highest-energy route. The challenge here is to find the best path that suits the rapid and secure deployment of data.

The least-cost algorithm and the best-energy path can be modeled and compared for a network to provide a behavioral benchmark. The model can determine all possible paths available between a given source and destination. The energy of each node and hence all possible least-cost paths are computed regularly by cluster heads and propagated to all cluster nodes for database updating.

During the phase of path finding to a cluster head, the routing algorithm accepts a failing (low-energy) node and finds the least-cost path, taking the failing node into account. The inputs for route process then include source node, destination node, failing nodes, and all other nodes.

13.4.2 Intercluster Routing Protocols

Intercluster protocols are not typically different from the multihop ones for intradomain cases. Interdomain protocols are available for

- Intercluster energy conscious routing (ICR)
- Energy-aware routing (EAR)
- Direct diffusion

ICR uses interest flooding similar to directed diffusion and EAR to establish routes between the base station and sensor nodes but differs from EAR and directed diffusion in some aspects.

13.4.2.1 Intercluster Energy-Conscious Routing (ICR)

ICR is a destination-initiated reactive routing protocol. This means that a destination, local base station (LBS), initiates an explicit route-discovery phase, which includes the propagation of an interest signal that floods throughout the network and establishes energy-efficient routes. Based on the application, which can be either periodic data collection or event driven, the interest signal can include the type and the period of the desired data as shown below.

For an application requiring information from specific locations, the interest signal also includes the position of the required information.

Type	Period	Sender's Address	Cost Field

Interest-signal structure in a packet

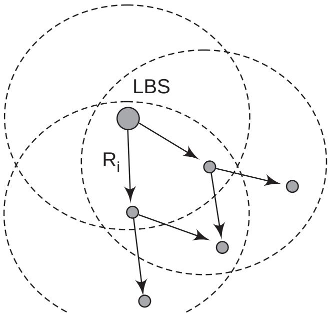


Figure 13.7: LBS starts route discovery by generating interest signals.

If the LBS requires some periodic data collection, it sets the period in which nodes send the specific type of information. Monitoring and surveillance applications are examples for the data-collection paradigm. If it requires sensor nodes to detect one specific event, an LBS includes the type of the event in the interest signal.

Following the route-discovery phase, sensor nodes switch to sleep mode and wait for the specific event. In case of event detection, non-cluster-head nodes send the data directly to the associated cluster head, which uses the previously established route to send the information back to the LBS. In short, ICR occurs in two phases: **route discovery** and **data acquisition**.

In route discovery, the local base station initiates route discovery by sending an interest signal within the range of R_i. The value of R_i should be both high enough to keep the cluster-head network connected and low enough to prevent unnecessary energy consumption and interest generation.

Owing to even distribution of cluster heads achieved by a clustering protocol, R_i can be chosen slightly bigger than the average distance between a pair of adjacent cluster heads. The LBS should adjust its output power and data rate of the interest signal to limit its transmission range to R_i. Also, the cost field is set to zero before interest propagation starts.

Since the distance between a pair of neighboring cluster heads is approximately the same across the network, the communication energy consumption associated with two distinct adjacent cluster heads is also the same. Therefore, the cost, or weight, of a multihop path is defined exclusively by the number of hops. In addition, the remaining energy in the cluster heads along the path affects the route-selection decision. The total-cost function C is defined as:

$$\mathrm{C} = ah + \beta \sum_{i} \frac{\mathrm{B_M}}{\mathrm{B}_{ri}}$$

where h is the hop number and B_{ri} represents the remaining energy in the battery of node i, B_M shows the maximum battery capacity of a sensor node, and α and β are normalization factors. The second part of the cost field favors the paths that include nodes with higher energy. To update the cost field, each intermediate cluster head calculates the inverse of its remaining battery power plus 1 (increment in the number of hops) and adds the outcome to the existing cost value.

The data-acquisition phase occurs after each cluster head collects the requested information from sensor nodes and compresses it into a packet with fixed length, searches for the neighbor's address in memory, and relays the packet to that neighbor. In order to reduce the diffusion of spare data bits in the network, relay nodes can receive the data packets, each of length L, from N nodes and aggregate them into one single packet of length L.

This reduces the number of data bits forwarded by the relay node from NL to L. To enable data aggregation during the data-collection period, cluster heads that are closer to the base station, that is, the cost field of the saved interest message includes fewer hops should wait for their neighbors to send their data packets and then compress the incoming information with their own data and send the packet with the fixed length to the relay neighbor.

13.4.2.2 Comparison of ICR and EAR

ICR is different from EAR in two aspects. In EAR, sensor nodes save and propagate most of the incoming interest signals and eliminate only the ones with a very high cost field. However, in ICR, every time that the cost field of the incoming interest message is higher than the previously saved one, the packet gets destroyed. This puts a limit on the generation of interest messages.

In EAR, in order to ensure that the optimal path does not get depleted and that the network degrades evenly, multiple paths are found between a source and a destination. Each node has to wait for all the interest signals to come and then calculates the average cost between itself and the destination. Based on the average cost, each path is assigned a probability of being chosen.

Depending on the probability, each time one of the paths is chosen, ICR assumes that data aggregation is executed among cluster heads, and no packet moves along the chosen path independently. This means that during the data-collection period, each cluster head aggregates the data from its N adjacent cluster heads and has to forward only one compressed packet rather than N distinct packets.

After the execution of the routing protocol, a spanning tree is established that is rooted in the base station and connects all the cluster heads to the base station. Therefore, only the least-cost, or the optimum, path is a final, established route for each cluster head. This way, the degradation of the optimal path for each packet is prevented.

13.5 OTHER RELATED TECHNOLOGIES

Other sensor-network based technologies use low-power nodes and we would be focusing on the ZigBee technology.

13.5.1 ZigBee Technology and IEEE 802.15.4

The ZigBee technology is a communication standard that provides a short-range low-cost networking capability that allows low-cost devices to quickly transmit small amounts of data, such as temperature readings for thermostats, on/off requests for light switches, or keystrokes for a wireless keyboard.

Other ZigBee applications are in professional installation kits for lighting controls, heating, ventilation, air-conditioning, and security. Even the Bluetooth short-range wireless technology found in laptops and cellphones lacks the affordability, power savings, and mesh-networking capabilities of ZigBee.

ZigBee comes from higher-layer enhancements by a multivendor consortium called the Zigbee Alliance. IEEE standard 802.15.4/ZigBee specifies the MAC and physical layers. The 802.15.4 standard specifies 128-bit AES encryption; ZigBee specifies how to handle encryption key exchange. The 802.15.4/ZigBee networks run in the unlicensed frequencies, 900 MHz and 2.4 GHz band, based on a packet radio standard and support many cordless telephones, allowing data to be sent over distances up to 20 meters.

ZigBee devices, typically battery powered, can transmit information much farther than 20 meters, because each device within listening distance passes the message along to any other device within range. Only the intended device acts on the message. By instructing nodes to wake up only for those split-second intervals when they are needed, ZigBee device batteries might last for years. Although this technology is targeting for manufacturing, health care, shipping, and Defence, the ZigBee Alliance is initially keeping its focus small.

SUMMARY

This final chapter focused on wireless sensor networks. Some applications of sensor networks are **target tracking, environmental monitoring, system control,** and **chemical or biological detection**.

The protocol stack for sensor networks concerned with power factor. The sensor network protocol stack combines two features: power-efficiency and least-cost-path routing. Thus, the protocol architecture integrates networking protocols and power efficiency through the wireless medium and promotes cooperative efforts among sensor nodes. The protocol stack consists of the physical, data-link, network, transport, and application layers, power-management, mobility-management, and task-management planes. The internal structure of an intelligent sensor node consists of three units for sensing, processing, and storage, respectively, and communications capabilities.

An energy model for a transceiver of a node can be developed. The energy consumption, E, for all components of a transceiver in watts can be modeled by $E = \theta + \eta\omega d^n$, where θ is the distance-independent term that accounts for the overhead of the radio electronics and digital processing, and $\eta\omega d^n$ is the distance-dependent term in which η represents the amplifier inefficiency factor, ω is the free-space path loss, and d is the distance.

Two clustering protocols in sensor networks are the Low-Energy Adaptive Clustering Hierarchy (LEACH) algorithm and the Decentralized Energy-Efficient Cluster Propagation (DEEP) protocol. DEEP is based on the idea of controlling the geographical dimensions of clusters and the distribution of cluster heads. Because of the balanced load among cluster heads, there is no need for frequent re-clustering, but after current cluster heads are out of energy, the protocol can rotate the cluster-head position among all the sensor nodes. Also the identical distance between a pair of neighboring cluster heads leads to the ease of route set-up deployment.

After establishing well-distributed cluster heads and clusters in a network, energy-conscious routing is essential in order to set communication routes among cluster heads. ZigBee technology, which is based on the IEEE 802.15.4 standard, is a related technology that uses low-power nodes.

INDEX

ABOUT THE BOOK

The book **"A Complete Guide to Computer Networks"** aims to provide an in-depth understanding on network technologies that have continued to revolutionize the world of communication, ever since. This book can be used as a textbook for the students of BCA, MCA as well as Computer Science branches of engineering.

The book can be characterized by the following features:

- It provides an in-depth of knowledge of the fundamentals of Computer Networks.
- It is an excellent guide for even beginners in this field.
- It also covers advanced topics, for those who may already be well-versed in the fundamentals.
- It uses a very easy language so that any average reader would also be able to benefit from the concepts discussed in the text.

ABOUT THE AUTHORS

Prof. Rachna Sharma is a keen educator, who is specializing in Computer Networks. Her areas of interest include Data and Computer Communication, Computer Networks and Wireless Technologies, on which she has previously written two books. She has been working as Head, Department of MCA at CMR Institute of Technology, Bangalore, since 2007. She is currently pursuing her Ph.D. in the field of E-Learning using 4G.

Prof. Sudipto Das has been in the field of Education & Training for above 10 years. He believes in diversification and has taught a wide range of subjects, to several categories of learners. He has previously taught Networks and Advanced Computer Networks. He has also written a book on Computer Fundamentals. He has been working in the Department of MCA, CMR Institute of Technology, Bangalore, since 2006. He is currently pursuing his Ph.D. in Mobile Learning for Rural Areas.

ISBN 978-93-81159-68-2

9 789381 159682

UCN-9715-195-A GUIDE TO COMPUTER NET-SHA

UNIVERSITY SCIENCE PRESS
(An Imprint of Laxmi Publications Pvt. Ltd.)
An ISO 9001:2008 Certified Company