SPCA 208

MASTER OF COMPUTER APPLICATIONS

SECOND YEAR

FOURTH SEMESTER

CORE PAPER - XVII

COMPUTER NETWORKS

INSTITUTE OF DISTANCE EDUCATION

UNIVERSITY OF MADRAS

MASTER OF COMPUTER APPLICATIONS
SECOND YEAR - FOURTH SEMESTER

CORE PAPER - XVII
COMPUTER NETWORKS

WELCOME

Warm Greetings.

It is with a great pleasure to welcome you as a student of Institute of Distance Education, University of Madras. It is a proud moment for the Institute of Distance education as you are entering into a cafeteria system of learning process as envisaged by the University Grants Commission. Yes, we have framed and introduced Choice Based Credit System(CBCS) in Semester pattern from the academic year 2018-19. You are free to choose courses, as per the Regulations, to attain the target of total number of credits set for each course and also each degree programme. What is a credit? To earn one credit in a semester you have to spend 30 hours of learning process. Each course has a weightage in terms of credits. Credits are assigned by taking into account of its level of subject content. For instance, if one particular course or paper has 4 credits then you have to spend 120 hours of self-learning in a semester. You are advised to plan the strategy to devote hours of self-study in the learning process. You will be assessed periodically by means of tests, assignments and quizzes either in class room or laboratory or field work. In the case of PG (UG), Continuous Internal Assessment for 20(25) percentage and End Semester University Examination for 80 (75) percentage of the maximum score for a course / paper. The theory paper in the end semester examination will bring out your various skills: namely basic knowledge about subject, memory recall, application, analysis, comprehension and descriptive writing. We will always have in mind while training you in conducting experiments, analyzing the performance during laboratory work, and observing the outcomes to bring out the truth from the experiment, and we measure these skills in the end semester examination. You will be guided by well experienced faculty.

I invite you to join the CBCS in Semester System to gain rich knowledge leisurely at your will and wish. Choose the right courses at right times so as to erect your flag of success. We always encourage and enlighten to excel and empower. We are the cross bearers to make you a torch bearer to have a bright future.

With best wishes from mind and heart,

DIRECTOR

MASTER OF COMPUTER APPLICATIONS
SECOND YEAR - FOURTH SEMESTER

CORE PAPER - XVII
COMPUTER NETWORKS

COURSE WRITER

Dr. R. Hema
Assistant Professor (T)
Department of Computer Science
IDE, University of Madras

EDITING AND CO-ORDINATION

Dr. S. Sasikala
Associate Professor in Computer Science
Institute of Distance Education
University of Madras
Chepauk, Chennnai - 600 005.

© UNIVERSITY OF MADRAS, CHENNAI 600 005.

MASTER OF COMPUTER APPLICATIONS

SECOND YEAR

FOURTH SEMESTER

Core Paper - XVII

COMPUTER NETWORKS

SYLLABUS

Objective ofthe course :

This course gives an insight into various network models and the generalnetwork design issues and related algorithms.

Unit1:

Introduction–NetworkHardware–Software–ReferenceModels–OSIandTCP/IP models – Example networks: Internet, 3G Mobile phone networks, Wireless LANs –RFID and sensor networks - Physical layer – Theoretical basis for data communication - guided transmissionmedia

Unit-2:

Wireless transmission - Communication Satellites – Digital modulation and multiplexing-Telephonesnetworkstructure–localloop,trunksandmultiplexing,switching. Data link layer: Design issues – error detection andcorrection.

Unit 3:

Elementary data link protocols - sliding window protocols – Example Data Link protocols – Packet over SONET, ADSL - Medium Access Layer – Channel Allocation Problem – Multiple Access Protocols.

Unit4:

Networklayer-designissues-Routingalgorithms-Congestioncontrolalgorithms– Quality of Service – Network layer of Internet- IP protocol – IP Address – Internet Control Protocol.

Unit 5:

Transport layer – transport service- Elements of transport protocol - Addressing, Establishing & Releasing a connection – Error control, flow control, multiplexing andcrash recovery-InternetTransportProtocol–TCP-NetworkSecurity:Cryptography.

Recommended Texts:

1) A. S. Tanenbaum, 2011, Computer Networks, Fifth Edition, Pearson Education, Inc.

Reference Books:

1) B.Forouzan,1998,IntroductiontoDataCommunicationsinNetworking,Tata McGraw Hill, NewDelhi.

2) F. Halsall, 1995, Data Communications, Computer Networks and Open Systems, AddisonWessley.

3) D. Bertsekas and R. Gallagher, 1992, Data Networks, Prentice hall of India, NewDelhi.

4) Lamarca,2002,CommunicationNetworks,TataMcGrawHill,NewDelhi.

Website, E-learning resources:

1) http://peasonhighered.com/tanenbaum

MASTER OF COMPUTER APPLICATIONS

SECOND YEAR

FOURTH SEMESTER

Core Paper - XVII

COMPUTER NETWORKS

SCHEME OF LESSONS

LESSON-1
BASICS OF COMPUTER NETWORKS

Structure

1.1 Introduction

Although the computer industry is still young compared to other industries (e.g., automobiles and air transportation), computers have made spectacular progress in a short time. During the first two decades of their existence, computer systems were highly centralized, usually within a single large room. A medium-sized company or university might have had one or two computers, while very large institutions had at most a few dozen. The idea that within forty years vastly more powerful computers smaller than postage stamps would be mass produced by the billions was pure science fiction.

The merging of computers and communications has had a profound influence on the way computer systems are organized. The old model of a single computer serving all of the organization's computational needs has been replaced by one in which a large number of separate but interconnected computers do the job. These systems are called computer networks. The design and organization of these networks are the subjects of this book.

1.2 Objectives

- To explain the basics of computer networks.
- To give an overview of Network Hardware and Network Software
- To explore the various types of networks.

1.3 Network Hardware

In computer networks, there are multiple devices or mediums which helps in the communication between two different devices which are known as Network devices. Ex: Repeater, Router, Switch, Hub, Bridge etc. Let us see the basic definitions for the network devices.

A **Repeater** operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

A **Hub** is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.Generally there are two types of Hub.

- **Active Hub:-** These are the hubs which have their own power supply and can clean, boost and relay the signal along with the network. It serves both as a repeater as well as wiring centre. These are used to extend the maximum distance between nodes.
- **Passive Hub :-** These are the hubs which collect wiring from nodes and power supply from active hub. These hubs relay signals onto the network without cleaning and boosting them and can't be used to extend the distance between nodes.

A Bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device. There are two types of bridges (i) Transparent bridges and (ii) Source routing bridges.

- **Transparent Bridges:-** These are the bridge in which the stations are completely unaware of the bridge's existence i.e. whether or not a bridge is added or deleted from the network, reconfiguration of the stations is unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.

- **Source Routing Bridges**:- These bridges rely on the source of the packet transmission to provide the bridge table information. The source computer determines the best path by sending out explorer packets. This information is used by the bridge to build its table. The destination determines the best route for future packets to take. The destination returns that information to the source. Token Ring networks mainly use source-routing bridges.

A **Switch** is a multiport bridge with a buffer and a design that can boost its efficiency(a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.

A **Router** is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it and it is explained in the Fig. 1.1.

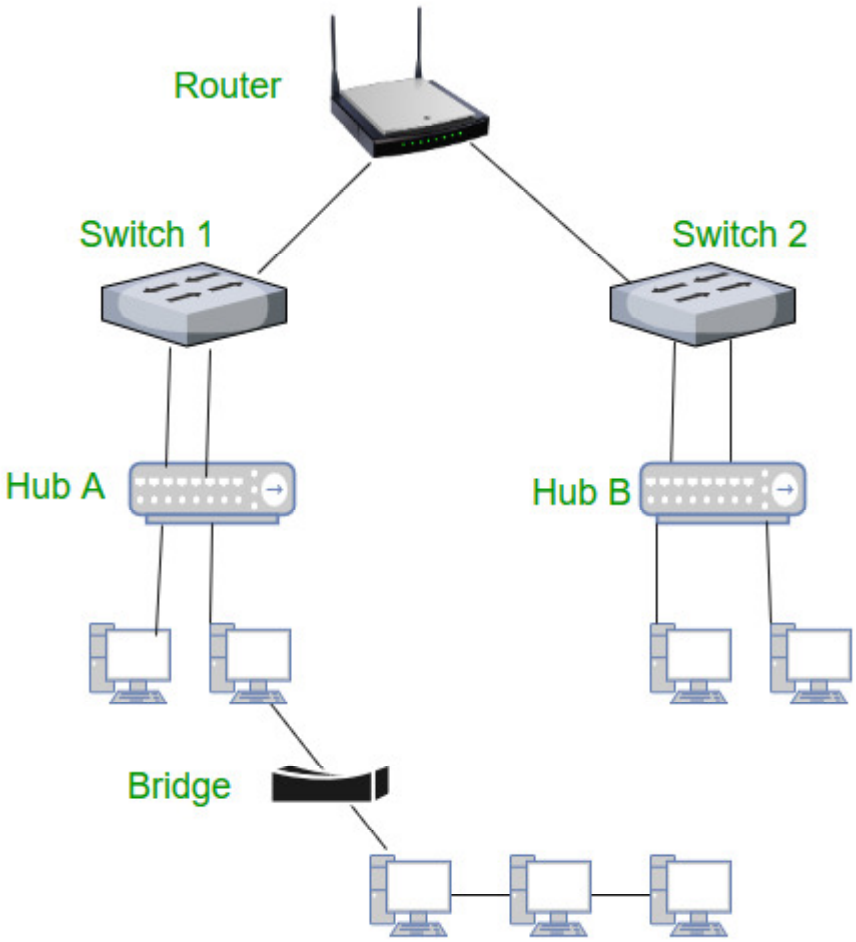


Figure 1.1 Division of hosts by routers

A **Gateway**, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.

Brouteris also known as bridging router, a device which combines features of both bridge and router. It can work either at data link layer or at network layer. Working as router, it is capable of routing packets across networks and working as bridge, it is capable of filtering local area network traffic.

It is now time to discuss about the technical issues involved in network design. There is no generally accepted taxonomy into which all computer networks fit, but two dimensions stand out as important: transmission technology and scale.

Broadly speaking, there are two types of transmission technology that are in widespread use: broadcast links and point-to-point links.

Point-to-point links connect individual pairs of machines. To go from the source to the destination on a network made up of point-to-point links, short messages, called packets in certain contexts, may have to first visit one or more intermediate machines. Often multiple routes, of different lengths, are possible, so finding good ones is important in point-to-point networks. Point-to-point transmission with exactly one sender and exactly one receiver is sometimes called **unicasting**.

In contrast, on a **broadcast network**, the communication channel is shared by all the machines on the network; packets sent by any machine are received by all the others. An address field within each packet specifies the intended recipient. Upon receiving a packet, a machine checks the address field. If the packet is intended for the receiving machine, that machine processes the packet; if the packet is intended for some other machine, it is just ignored.

A wireless network is a common example of a broadcast link, with communication shared over a coverage region that depends on the wireless channel and the transmitting machine. As an analogy, consider someone standing in a meeting room and shouting "Watson, come here. I want you." Although the packet may actually be received (heard) by many people, only Watson will respond; the others just ignore it.

Broadcast systems usually allow the possibility of addressing a packet to all destinations by using a special code in the address field. When a packet with this code is transmitted, it is received and processed by every machine on the network. This mode of operation is called **broadcasting**. Some broadcast systems also support transmission to a subset of the machines, which known as **multicasting**.

1.3.1 Types of Networks

An alternative criterion for classifying networks is by scale. Distance is important as a classification metric because different technologies are used at different scales.

In Fig. 1.2 we classify multiple processor systems by their rough physical size. At the top are the personal area networks, networks that are meant for one person. Beyond these come longer-range networks. These can be divided into local, metropolitan, and wide area networks, each with increasing scale. Finally, the connection of two or more networks is called an **internetwork**. The worldwide Internet is certainly the best-known (but not the only) example of an internetwork. In the following sections, we give a brief introduction to network hardware by scale.

Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

Figure 1.2 Classification of interconnected processors by scale

Personal Area Networks

A P**ersonal Area Network** (PAN) is a **computer network** for interconnecting devices centered on an individual person's workspace. A common example is a wireless network that connects a computer with its peripherals. Almost every computer has an attached monitor, keyboard, mouse, and printer. Without using wireless, this connection must be done with cables.

So many new users have a hard time finding the right cables and plugging them into the right little holes that most computer vendors offer the option of sending a technician to the user's home to do it. To help these users, some companies got together to design a short-range wireless network called Bluetooth to connect these components without wires. The idea is that if your devices have Bluetooth, then you need no cables. You just put them down, turn them on, and they work together. For many people, this ease of operation is a big plus.

In the simplest form, Bluetooth networks use the master-slave paradigm of Fig. 1.3. The system unit (the PC) is normally the master, talking to the mouse, keyboard, etc., as slaves. The master tells the slaves what addresses to use, when they can broadcast, how long they can transmit, what frequencies they can use, and so on.

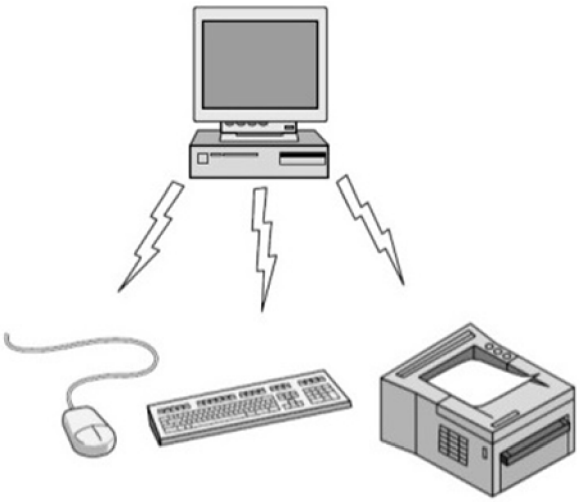


Figure 1.3 Bluetooth PAN configuration

Bluetooth can be used in other settings, too. It is often used to connect a headset to a mobile phone without cords and it can allow your digital music player to connect to your car merely being brought within range. A completely different kind of PAN is formed when an embedded medical device such as a pacemaker, insulin pump, or hearing aid talks to a user-operated remote control.

PANs can also be built with other technologies that communicate over short ranges, such as RFID on smartcards and library books.

Local Area Networks

The next step up is the LAN (Local Area Network). A LAN is a privately owned network that operates within and nearby a single building like a home, office or factory. LANs are widely used to connect personal computers and consumer electronics to let them share resources (e.g., printers) and exchange information. When LANs are used by companies, they are called **enterprise networks**.

Wireless LANs are very popular these days, especially in homes, older office buildings, cafeterias, and other places where it is too much trouble to install cables. In these systems, every computer has a radio modem and an antenna that it uses to communicate with other computers. In most cases, each computer talks to a device in the ceiling as shown in Fig. 1.4(a). This device, called an AP (Access Point), wireless router, or base station, relays packets between the wireless computers and also between them and the Internet. Being the AP is like being the popular kid as school because everyone wants to talk to you. However, if other computers are close enough, they can communicate directly with one another in a peer-to-peer configuration.

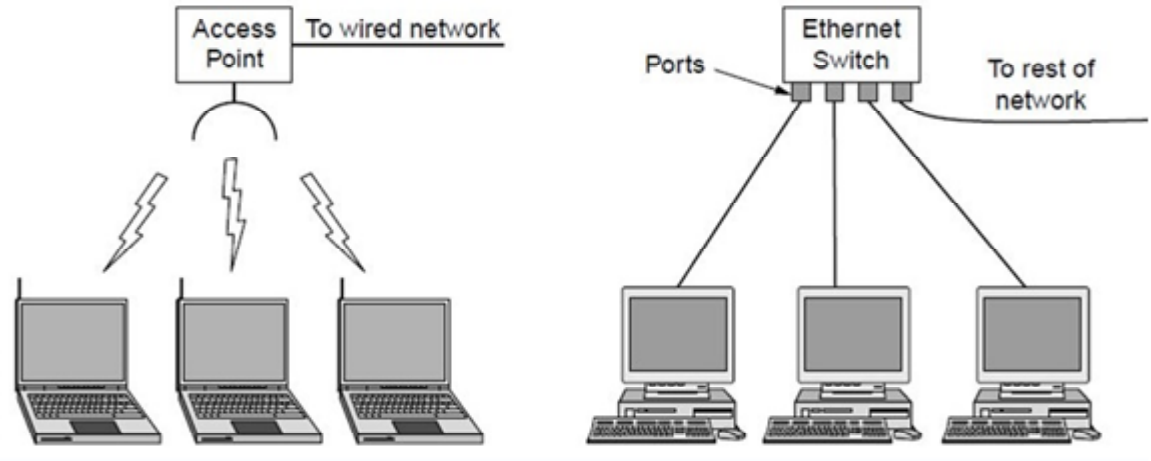


Figure 1.4 Wireless and wired LANs. (a) 802.11. (b) Switched Ethernet

Wired LANs use a range of different transmission technologies. Most of them use copper wires, but some use optical fiber. LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. Knowing these bounds helps with the task of designing network protocols. Typically, wired LANs run at speeds of 100 Mbps

to 1 Gbps, have low delay (microseconds or nanoseconds), and make very few errors. Newer LANs can operate at up to 10 Gbps. Compared to wireless networks, wired LANs exceed them in all dimensions of performance. It is just easier to send signals over a wire or through a fiber than through the air.

The topology of many wired LANs is built from point-to-point links. IEEE 802.3, popularly called Ethernet, is, by far, the most common type of wired LAN. Fig. 1-4(b) shows a sample topology of switched Ethernet. Each computer speaks the Ethernet protocol and connects to a box called a switch with a point-to-point link. Hence the name. A switch has multiple ports, each of which can connect to one computer. The job of the switch is to relay packets between computers that are attached to it, using the address in each packet to determine which computer to send it to.

In the future, it is likely that every appliance in the home will be capable of communicating with every other appliance, and all of them will be accessible over the Internet. Many devices are already capable of being networked. These include computers, entertainment devices such as TVs and DVDs, phones and other consumer electronics such as cameras, appliances like clock radios, and infrastructure like utility meters and thermostats. This trend will only continue. For instance, the average home probably has a dozen clocks, all of which could adjust to daylight savings time automatically if the clocks were on the Internet. Remote monitoring of the home is a likely winner, as many grown children would be willing to spend some money to help their aging parents live safely in their own homes.

In short, home LANs offer many opportunities and challenges. Most of the challenges relate to the need for the networks to be easy to manage, dependable, and secure, especially in the hands of nontechnical users, as well as low cost.

Metropolitan Area Networks

A MAN (Metropolitan Area Network) covers a city. The best-known examples of MANs are the cable television networks available in many cities. These systems grew from earlier community antenna systems used in areas with poor over-the-air television reception. In those early systems, a large antenna was placed on top of a nearby hill and a signal was then piped to the subscribers' houses.

At first, these were locally designed, ad hoc systems. Then companies began jumping into the business, getting contracts from local governments to wire up entire cities. The next step was television programming and even entire channels designed for cable only. Often these channels were highly specialized, such as all news, all sports, all cooking, all gardening, and so on. But from their inception until the late 1990s, they were intended for television reception only.

When the Internet began attracting a mass audience, the cable TV network operators began to realize that with some changes to the system, they could provide two-way Internet service in unused parts of the spectrum. At that point, the cable TV system began to morph from simply a way to distribute television to a metropolitan area network. To a first approximation, a MAN might look something like the system shown in Fig. 1.5. In this figure we see both television signals and Internet being fed into the centralized cable headend for subsequent distribution to people's homes.

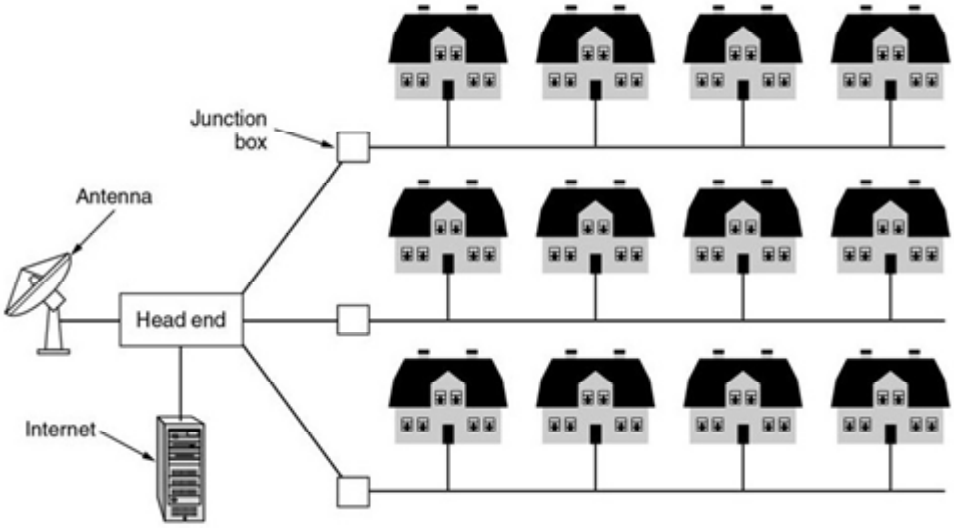


Figure 1.5 A metropolitan area network based on cable TV

Cable television is not the only MAN, though. Recent developments in highspeed wireless Internet access have resulted in another MAN, which has been standardized as IEEE 802.16 and is popularly known as WiMAX.

1.3.4 Wide Area Networks

A WAN (Wide Area Network) spans a large geographical area, often a country or continent. We will begin our discussion with wired WANs, using the example of a company with branch offices in different cities.

The WAN in Fig. 1.6 is a network that connects offices in Perth, Melbourne, and Brisbane. Each of these offices contains computers intended for running user (i.e., application) programs. We will follow traditional usage and call these machines hosts. The rest of the network that connects these hosts is then called the communication subnet, or just subnet for short. The job of the subnet is to carry messages from host to host, just as the telephone system carries words (really just sounds) from speaker to listener.

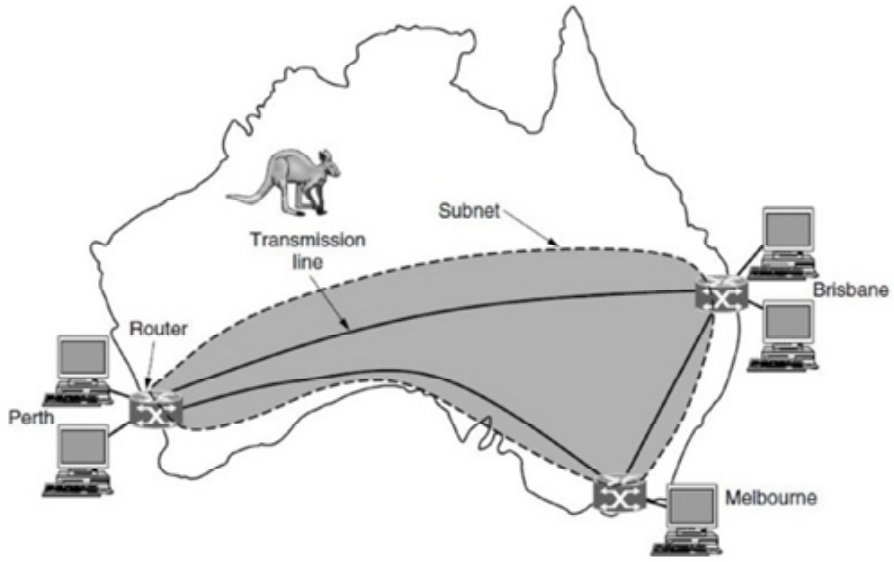


Figure 1.6 WAN that connects three branch offices in Australia

In most WANs, the subnet consists of two distinct components: transmission lines and switching elements. Transmission lines move bits between machines. They can be made of copper wire, optical fiber, or even radio links. Most companies do not have transmission lines lying about, so instead they lease the lines from a telecommunications company. Switching elements, or just switches, are specialized computers that connect two or more transmission lines. When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them. These switching computers have been called by various names in the past; the name router is now most commonly used.

In WANs, the network contains many transmission lines, each connecting a pair of routers. If two routers that do not share a transmission line wish to communicate, they must do this indirectly, via other routers. There may be many paths in the network that connect these two

routers. How the network makes the decision as to which path to use is called the routing algorithm. Many such algorithms exist. How each router makes the decision as to where to send a packet next is called the forwarding algorithm. Many of them exist too.

Other kinds of WANs make heavy use of wireless technologies. In satellite systems, each computer on the ground has an antenna through which it can send data to and receive data from to a satellite in orbit. All computers can hear the output from the satellite, and in some cases they can also hear the upward transmissions of their fellow computers to the satellite as well. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

Internetworks

Many networks exist in the world, often with different hardware and software. People connected to one network often want to communicate with people attached to a different one. The fulfillment of this desire requires that different, and frequently incompatible, networks be connected. A collection of interconnected networks is called an **internetwork** or **internet**. These terms will be used in a generic sense, in contrast to the worldwide Internet (which is one specific internet), which we will always capitalize. The Internet uses ISP networks to connect enterprise networks, home networks, and many other networks.

Subnets, networks, and internetworks are often confused. The term "subnet" makes the most sense in the context of a wide area network, where it refers to the collection of routers and communication lines owned by the network operator. As an analogy, the telephone system consists of telephone switching offices connected to one another by high-speed lines, and to houses and businesses by low-speed lines. These lines and equipment, owned and managed by the telephone company, form the subnet of the telephone system.

We know that an internet is formed when distinct networks are interconnected. In our view, connecting a LAN and a WAN or connecting two LANs is the usual way to form an internetwork, but there is little agreement in the industry over terminology in this area. There are two rules of thumb that are useful. First, if different organizations have paid to construct different parts of the network and each maintains its part, we have an internetwork rather than

a single network. Second, if the underlying technology is different in different parts (e.g., broadcast versus point-to-point and wired versus wireless), we probably have an internetwork.

To go deeper, the general name for a machine that makes a connection between two or more networks and provides the necessary translation, both in terms of hardware and software, is a gateway. Gateways are distinguished by the layer at which they operate in the protocol hierarchy.

Since the benefit of forming an internet is to connect computers across networks, we do not want to use too low-level a gateway or we will be unable to make connections between different kinds of networks. We do not want to use too high-level a gateway either, or the connection will only work for particular applications. The level in the middle that is "just right" is often called the network layer, and a router is a gateway that switches packets at the network layer.

1.4 Network Software

The first computer networks were designed with the hardware as the main concern and the software as an afterthought. This strategy no longer works. Network software is now highly structured. In the following sections we examine the software structuring technique in some detail.

1.4.1 Protocol Hierarchies

To reduce their design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it.

This concept is actually a familiar one and is used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware)

provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

When layer *n* on one machine carries on a conversation with layer n on another machine, the rules and conventions used in this conversation are collectively known as the layer n protocol. Basically, a protocol is an agreement between the communicating parties on how communication is to proceed.

A five-layer network is illustrated in Fig. 1.7. The entities comprising the corresponding layers on different machines are called **peers**. The peers may be software processes, hardware devices, or even human beings. In other words, it is the peers that communicate by using the protocol to talk to each other.

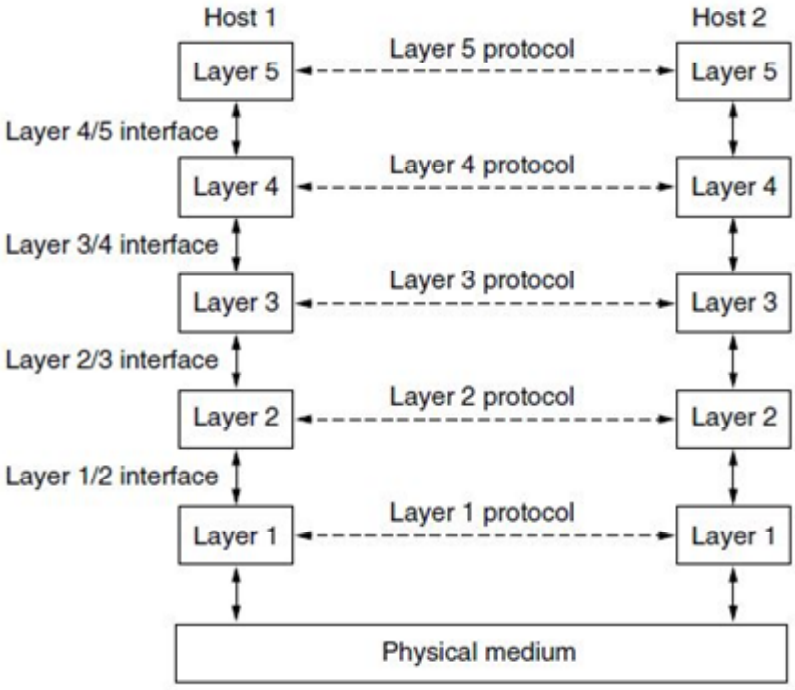


Figure 1.7 Layers, protocols, and interfaces

In reality, no data are directly transferred from layer n on one machine to layer n on another machine. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached. Below layer 1 is the physical medium through which actual communication occurs. In Fig. 1.1, virtual communication is shown by dotted lines and physical communication by solid lines.

Between each pair of adjacent layers is an interface. The interface defines which primitive operations and services the lower layer makes available to the upper one. When network designers decide how many layers to include in a network and what each one should do, one of the most important considerations is defining clean interfaces between the layers. Doing so, in turn, requires that each layer perform a specific collection of well-understood functions. In addition to minimizing the amount of information that must be passed between layers, clearcut interfaces also make it simpler to replace one layer with a completely different protocol or implementation because all that is required of the new protocol or implementation is that it offer exactly the same set of services to its upstairs neighbor as the old one did. It is common that different hosts use different implementations of the same protocol. In fact, the protocol itself can change in some layer without the layers above and below it even noticing.

A set of layers and protocols is called a **network architecture**. The specification of an architecture must contain enough information to allow an implementer to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol. Neither the details of the implementation nor the specification of the interfaces is part of the architecture because these are hidden away inside the machines and not visible from the outside. It is not even necessary that the interfaces on all machines in a network be the same, provided that each machine can correctly use all the protocols. A list of the protocols used by a certain system, one protocol per layer, is called a protocol stack.

An analogy may help explain the idea of multilayer communication. Imagine two philosophers (peer processes in layer 3), one of whom speaks Urdu and English and one of whom speaks Chinese and French. Since they have no common language, they each engage a translator (peer processes at layer 2), each of whom in turn contacts a secretary (peer processes in layer 1). Philosopher 1 wishes to convey his affection to his peer. To do so, he passes a message (in English) across the 2/3 interface to his translator, saying "I like rabbits,". The translators have agreed on a neutral language known to both of them, Dutch, so the message is converted to "Ikvindkonijnenleuk." The choice of the language is the layer 2 protocol and is up to the layer 2 peer processes.

The translator then gives the message to a secretary for transmission, for example, by email (the layer 1 protocol). When the message arrives at the other secretary, it is passed to the

local translator, who translates it into French and passes it across the 2/3 interface to the second philosopher. Note that each protocol is completely independent of the other ones as long as the interfaces are not changed. The translators can switch from Dutch to, say, Finnish, at will, provided that they both agree and neither changes his interface with either layer 1 or layer 3. Similarly, the secretaries can switch from email to telephone without disturbing (or even informing) the other layers. Each process may add some information intended only for its peer. This information is not passed up to the layer above.

The peer process abstraction is crucial to all network design. Using it, the unmanageable task of designing the complete network can be broken into several smaller, manageable design problems, namely, the design of the individual layers.

1.4.2 Connection-Oriented Versus Connectionless Service

Layers can offer two different types of service to the layers above them: **connection-oriented** and **connectionless**. In this section we will look at these two types and examine the differences between them.

Connection-oriented service is modeled after the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up. Similarly, to use a connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection. The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end. In most cases the order is preserved so that the bits arrive in the order they were sent.

In some cases when a connection is established, the sender, receiver, and subnet conduct a negotiation about the parameters to be used, such as maximum message size, quality of service required, and other issues. Typically, one side makes a proposal and the other side can accept it, reject it, or make a counterproposal. A circuit is another name for a connection with associated resources, such as a fixed bandwidth. This dates from the telephone network in which a circuit was a path over copper wire that carried a phone conversation.

In contrast to connection-oriented service, connectionless service is modeled after the postal system. Each message carries the full destination address,and each one is routed through the intermediate nodes inside the system independent of all the subsequent messages. There are different names for messages in different contexts; a packet is a message at the network layer. When the intermediate nodes receive a message in full before sending it on to the next node, this is called **store-and-forward switching**. The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called **cut-through** switching. Normally, when two messages are sent to the same destination, the first one sent will be the first one to arrive. However, it is possible that the first one sent can be delayed so that the second one arrives first.

A typical situation in which a reliable connection-oriented service is appropriate is file transfer. The owner of the file wants to be sure that all the bits arrive correctly and in the same order they were sent. Very few file transfer customers would prefer a service that occasionally scrambles or loses a few bits, even if it is much faster.

Not all applications require connections. For example, spammers send electronic junk-mail to many recipients. The spammer probably does not want to go to the trouble of setting up and later tearing down a connection to a recipient just to send them one item. Nor is 100 percent reliable delivery essential, especially if it costs more. All that is needed is a way to send a single message that has a high probability of arrival, but no guarantee. Unreliable (meaning not acknowledged) connectionless service is often called **datagram service**, in analogy with telegram service, which also does not return an acknowledgement to the sender. Despite it being unreliable, it is the dominant form in most networks for reasons that will become clear later.

In other situations, the convenience of not having to establish a connection to send one message is desired, but reliability is essential. The acknowledged datagram service can be provided for these applications. It is like sending a registered letter and requesting a return receipt. When the receipt comes back, the sender is absolutely sure that the letter was delivered to the intended party and not lost along the way. Text messaging on mobile phones is an example.

Still another service is the request-reply service. In this service the sender transmits a single datagram containing a request; the reply contains the answer. Request-reply is commonly used to implement communication in the client-server model: the client issues a request and the server responds to it. For example, a mobile phone client might send a query to a map server to retrieve the map data for the current location.

1.4.3 Service Primitives

A service is formally specified by a set of primitives (operations) available to user processes to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, as it often is, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets. The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connectionless service. As a minimal example of the service primitives that might provide a reliable byte stream, consider the primitives listed in Fig. 1.8. They will be familiar to fans of the Berkeley socket interface, as the primitives are a simplified version of that interface.

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
ACCEPT	Accept an incoming connection from a peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Figure 1.8 Six service primitives that provide a simple connection-oriented service

These primitives might be used for a request-reply interaction in a client-server environment. To illustrate how, we sketch a simple protocol that implements the service using acknowledged datagrams.

First, the server executes LISTEN to indicate that it is prepared to accept incoming connections. A common way to implement LISTEN is to make it a blocking system call. After executing the primitive, the server process is blocked until a request for connection appears.

Next, the client process executes CONNECT to establish a connection with the server. The CONNECT call needs to specify who to connect to, so it might have a parameter giving the server's address. The operating system then typically sends a packet to the peer asking it to connect, as shown by (1) in Fig. 1.9. The client process is suspended until there is a response.

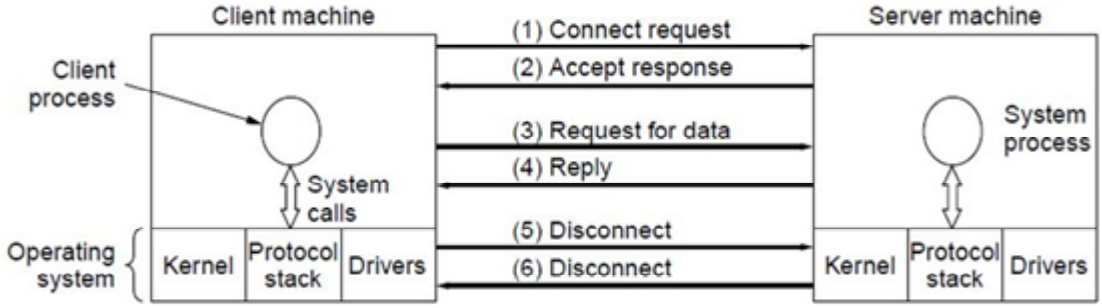


Figure 1.9 A simple client-server interaction using acknowledged datagrams

When the packet arrives at the server, the operating system sees that the packet is requesting a connection. It checks to see if there is a listener, and if so it unblocks the listener. The server process can then establish the connection with the ACCEPT call. This sends a response (2) back to the client process to accept theconnection. The arrival of this response then releases the client. At this point the client and server are both running and they have a connection established.

The obvious analogy between this protocol and real life is a customer (client) calling a company's customer service manager. At the start of the day, the service manager sits next to his telephone in case it rings. Later, a client places a call. When the manager picks up the phone, the connection is established.

The next step is for the server to execute RECEIVE to prepare to accept the first request. Normally, the server does this immediately upon being released from the LISTEN, before the acknowledgement can get back to the client. The RECEIVE call blocks the server.

Then the client executes SEND to transmit its request (3) followed by the execution of RECEIVE to get the reply. The arrival of the request packet at the server machine unblocks the server so it can handle the request. After it has done the work, the server uses SEND to return

the answer to the client (4). The arrival of this packet unblocks the client, which can now inspect the answer. If the client has additional requests, it can make them now.

When the client is done, it executes DISCONNECT to terminate the connection (5). Usually, an initial DISCONNECT is a blocking call, suspending the client and sending a packet to the server saying that the connection is no longer needed. When the server gets the packet, it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection (6). When the server's packet gets back to the client machine, the client process is released and the connection is broken. In a nutshell, this is how connection-oriented communication works.

Many things can go wrong here. The timing can be wrong (e.g., the CONNECT is done before the LISTEN), packets can get lost, and much more. Fig. 1-9 briefly summarizes how client-server communication might work with acknowledged datagrams so that we can ignore lost packets. In short, in the real world, a simple request-reply protocol over an unreliable network is often inadequate. There are a variety of protocols that overcome the above problems. For the moment, suffice it to say that having a reliable, ordered byte stream between processes is sometimes very convenient.

1.4.4 The Relationship of Services to Protocols

Services and protocols are distinct concepts. A service is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.

A protocol, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users. In this way, the service and the protocol are completely decoupled. This is a key concept that any network designer should understand well.

To repeat this crucial point, services relate to the interfaces between layers, as illustrated in Fig. 1.10. In contrast, protocols relate to the packets sent between peer entities on different machines. It is very important not to confuse the two concepts.

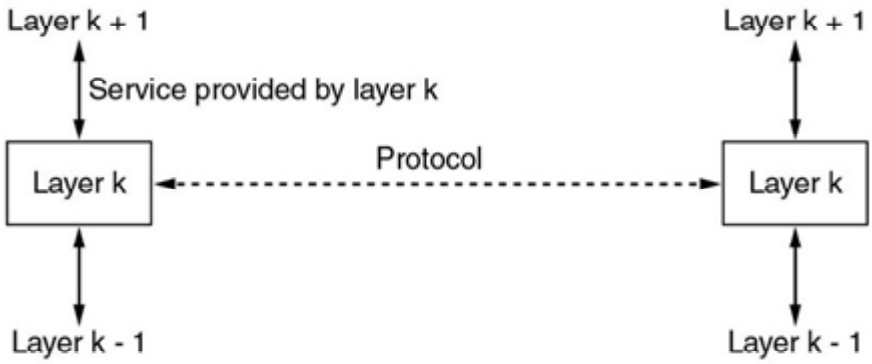


Figure 1.10 The relationship between a service and a protocol

An analogy with programming languages is worth making. A service is like an abstract data type or an object in an object-oriented language. It defines operations that can be performed on an object but does not specify how these operations are implemented. In contrast, a protocol relates to the implementation of the service and as such is not visible to the user of the service.

1.5 Summary

Computer networks have many uses, both for companies and for individuals, in the home and while on the move. Companies use networks of computers to share corporate information. For individuals, networks offer access to a variety of information and entertainment resources, as well as a way to buy and sell products and services. Technology advances are enabling new kinds of mobile applications and networks with computers embedded in appliances and other consumer devices. The same advances raise social issues such as privacy concerns.

Roughly speaking, networks can be divided into LANs, MANs, WANs, and internetworks. Networks can be interconnected with routers to form internetworks, of which the Internet is the largest and best known example. Wireless networks are also becoming extremely popular. Network software is built around protocols, which are rules by which processes communicate.

Networks provide various services to their users. These services can range from connectionless best-efforts packet delivery to connection-oriented guaranteed delivery. In some networks, connectionless service is provided in one layer and connection-oriented service is provided in the layer above it.

1.6 Model Questions

1. Discuss on Network Hardware.
2. Write notes on WAN.
3. Give a detailed note on wireless networks.
4. What are the benefits and requirements of computer networks?
5. Define the terms: (i) Gateway (ii) Bridge (iii) Router
6. Differentiate between connection-oriented and connectionless services?

LESSON-2
EVOLUTION OF NETWORKS

Structure

2.1 Introduction

The subject of computer networking covers many different kinds of networks, large and small, well known and less well known. They have different goals, scales, and technologies. In the following sections, we will look at some examples, to get an idea of the variety one finds in the area of computer networking.

We will start with the Internet, probably the best known network, and look at its history, evolution, and technology. Then we will consider the mobile phone network. Technically, it is quite different from the Internet, contrasting nicely with it. Next we will introduce IEEE 802.11, the dominant standard for wireless LANs. Finally, we will look at RFID and sensor networks, technologies that extend the reach of the network to include the physical world and everyday objects.

2.2 Objectives

- To explain the different examples in networks.
- To discuss about the evolution of the internet and it's architecture.
- To explore the concepts in 3G mobile phone networks, Wireless LANs, RFID and Sensor Networks

2.3 The Internet

The Internet is not really a network at all, but a vast collection of different networks that use certain common protocols and provide certain common services. It is an unusual system in that it was not planned by anyone and is not controlled by anyone. To better understand it, let us start from the beginning and see how it has developed and why.

2.3.1 The ARPANET

The ARPANET was a project funded by the U.S. government during the Cold War, in order to build a robust and reliable communications network. This was done by connecting various computers that could simultaneously communicate in a network that would not go down and continue running when a single node was taken out.

ARPANET was the network that became the basis for the Internet. Based on a concept first published in 1967, ARPANET was developed under the direction of the U.S. Advanced Research Projects Agency (ARPA). In 1969, the idea became a modest reality with the interconnection of four university computers. The initial purpose was to communicate with and share computer resources among mainly scientific users at the connected institutions. ARPANET took advantage of the new idea of sending information in small units called packets that could be routed on different paths and reconstructed at their destination. The development of the TCP/IP protocols in the 1970s made it possible to expand the size of the network, which now had become a network of networks, in an orderly way.

In the 1980s, ARPANET was handed over to a separate new military network, the Defense Data Network, and NSFNet, a network of scientific and academic computers funded by the National Science Foundation. In 1995, NSFNet in turn began a phased withdrawal to turn the

backbone of the Internet (called vBNS) over to a consortium of commercial backbone providers (PSINet, UUNET,ANS/AOL, Sprint, MCI, and AGIS-Net99).

Because ARPA's name was changed to Defense Advanced Research Projects Agency (DARPA) in 1971, ARPANET is sometimes referred to as DARPANET. (DARPA was changed back to ARPA in 1993 and back to DARPA again in 1996.)

The original ARPANET design is shown in Fig. 1-26.The software was split into two parts: subnet and host. The subnet software consisted of the IMP (Interface Message Processor)end of the host-IMP connection, the IMP-IMP protocol, and a source IMP to destination IMP protocol designed to improve reliability.

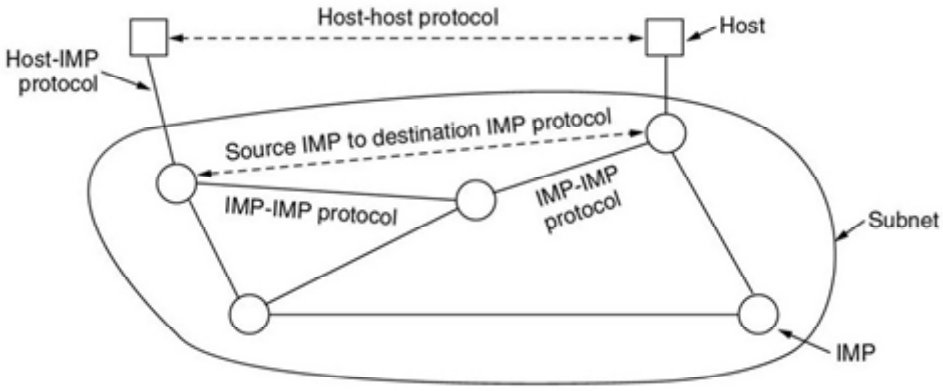


Figure 2.1 The original ARPANET design

Outside the subnet, software was also needed, namely, the host end of the host-IMP connection, the host-host protocol, and the application software. It soon became clear that BBN was of the opinion that when it had accepted a message on a host-IMP wire and placed it on the host-IMP wire at the destination, its job was done.

In addition to helping the fledgling ARPANET grow, ARPA also funded research on the use of satellite networks and mobile packet radio networks. In one famous demonstration, a truck driving around in California used the packet radio network to send messages to SRI, which were then forwarded over the ARPANET to the East Coast, where they were shipped to University College in London over the satellite network. This allowed a researcher in the truck to use a computer in London while driving around in California.

This experiment also demonstrated that the existing ARPANET protocols were not suitable for running over different networks. This observation led to more research on protocols, culminating with the invention of the TCP/IP model and protocols (Cerf and Kahn, 1974). TCP/IP was specifically designed to handle communication over internetworks, something becoming increasingly important as more and more networks were hooked up to the ARPANET.

To encourage adoption of these new protocols, ARPA awarded several contracts to implement TCP/IP on different computer platforms, including IBM, DEC, and HP systems, as well as for Berkeley UNIX. Researchers at the University of California at Berkeley rewrote TCP/IP with a new programming interface called sockets for the upcoming 4.2BSD release of Berkeley UNIX. They also wrote many application, utility, and management programs to show how convenient it was to use the network with sockets.

The timing was perfect. Many universities had just acquired a second or third VAX computer and a LAN to connect them, but they had no networking software. When 4.2BSD came along, with TCP/IP, sockets, and many network utilities, the complete package was adopted immediately. Furthermore, with TCP/IP, it was easy for the LANs to connect to the ARPANET, and many did.

During the 1980s, additional networks, especially LANs, were connected to the ARPANET. As the scale increased, finding hosts became increasingly expensive, so DNS (Domain Name System) was created to organize machines into domains and map host names onto IP addresses. Since then, DNS has become a generalized, distributed database system for storing a variety of information related to naming.

2.3.2 NSFNET

By the late 1970s, NSF (the U.S. National Science Foundation) saw the enormous impact the ARPANET was having on university research, allowing scientists across the country to share data and collaborate on research projects. However, to get on the ARPANET a university had to have a research contract with the DoD.Many did not have a contract. NSF's initial response was to fund the Computer Science Network (CSNET) in 1981. It connected computer science departments and industrial research labs to the ARPANET via dial-up and leased lines. In the late 1980s, the NSF went further and decided to design a successor to the ARPANET that would be open to all university research groups.

To have something concrete to start with, NSF decided to build a backbone network to connect its six supercomputer centers, in San Diego, Boulder, Champaign, Pittsburgh, Ithaca, and Princeton. Each supercomputer was given a little brother, consisting of an LSI-11 microcomputer called a fuzzball. The fuzzballs were connected with 56-kbps leased lines and formed the subnet, the same hardware technology the ARPANET used. The software technology was different however: the fuzzballs spoke TCP/IP right from the start, making it the first TCP/IP WAN.

NSF also funded some (eventually about 20) regional networks that connected to the backbone to allow users at thousands of universities, research labs, libraries, and museums to access any of the supercomputers and to communicate with one another. The complete network, including backbone and the regional networks, was called NSFNET. It connected to the ARPANET through a link between an IMP and a fuzzball in the Carnegie-Mellon machine room. The first NSFNET backbone is illustrated in Fig. 2-2 superimposed on a map of the U.S.

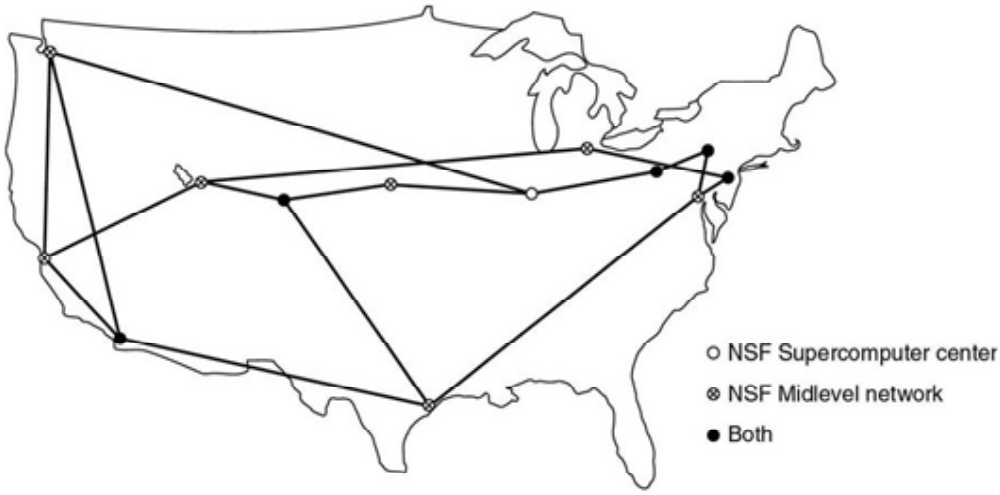


Figure 2.2 : The NSFNET backbone in 1988

NSFNET was an instantaneous success and was overloaded from the word go. NSF immediately began planning its successor and awarded a contract to the Michigan-based MERIT consortium to run it. Fiber optic channels at 448 kbps were leased from MCI (since merged with WorldCom) to provide the version 2 backbone. IBM PC-RTs were used as routers. This, too, was soon overwhelmed, and by 1990, the second backbone was upgraded to 1.5 Mbps.

As growth continued, NSF realized that the government could not continue financing networking forever. Furthermore, commercial organizations wanted to join but were forbidden by NSF's charter from using networks NSF paid for. Consequently, NSF encouraged MERIT, MCI, and IBM to form a nonprofit corporation, ANS (Advanced Networks and Services), as the first step along the road to commercialization. In 1990, ANS took over NSFNET and upgraded the 1.5-Mbps links to 45 Mbps to form ANSNET. This network operated for 5 years and was then sold to America Online. But by then, various companies were offering commercial IP service and it was clear the government should now get out of the networking business.

To ease the transition and make sure every regional network could communicate with every other regional network, NSF awarded contracts to four different network operators to establish a NAP (Network Access Point). These operators were PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.), and Sprint (New York City, where for NAP purposes, Pennsauken, New Jersey counts as New York City). Every network operator that wanted to provide backbone service to the NSF regional networks had to connect to all the NAPs.

This arrangement meant that a packet originating on any regional network had a choice of backbone carriers to get from its NAP to the destination's NAP. Consequently, the backbone carriers were forced to compete for the regional networks' business on the basis of service and price, which was the idea, of course. As a result, the concept of a single default backbone was replaced by a commercially driven competitive infrastructure. Many people like to criticize the Federal Government for not being innovative, but in the area of networking, it was DoD and NSF that created the infrastructure that formed the basis for the Internet and then handed it over to industry to operate.

During the 1990s, many other countries and regions also built national research networks, often patterned on the ARPANET and NSFNET. These included EuropaNET and EBONE in Europe, which started out with 2-Mbps lines and then upgraded to 34-Mbps lines. Eventually, the network infrastructure in Europe was handed over to industry as well.

The Internet has changed a great deal since those early days. It exploded in size with the emergence of the World Wide Web (WWW) in the early 1990s. Recent data from the

Internet Systems Consortium puts the number of visible Internet hosts at over 600 million. This guess is only a low-ball estimate, but it far exceeds the few million hosts that were around when the first conference on the WWW was held at CERN in 1994.

The way we use the Internet has also changed radically. Initially, applications such as email-for-academics, newsgroups, remote login, and file transfer dominated. Later it switched to email-for-everyman, then the Web and peer-to-peer content distribution, such as the now-shuttered Napster. Now real-time media distribution, social networks (e.g., Facebook), and microblogging (e.g., Twitter) are taking off. These switches brought richer kinds of media to the Internet and hence much more traffic. In fact, the dominant traffic on the Internet seems to change with some regularity as, for example, new and better ways to work with music or movies can become very popular very quickly.

2.3.3 Architecture of the Internet

The architecture of the Internet has also changed a great deal as it has grown explosively. In this section, we will attempt to give a brief overview of what it looks like today. The picture is complicated by continuous upheavals in the businesses of telephone companies (telcos), cable companies and ISPs that often make it hard to tell who is doing what. One driver of these upheavals is telecommunications convergence, in which one network is used for previously different uses. For example, in a "triple play" one company sells you telephony, TV, and Internet service over the same network connection on the assumption that this will save you money. Consequently, the description given here will be of necessity somewhat simpler than reality. And what is true today may not be true tomorrow.

The big picture is shown in Fig. 2.3. Let us examine this figure piece by piece, starting with a computer at home (at the edges of the figure). To join the Internet, the computer is connected to an Internet Service Provider, or simply ISP, from who the user purchases Internet access or connectivity. This lets the computer exchange packets with all of the other accessible hosts on the Internet. The user might send packets to surf the Web or for any of a thousand other uses, it does not matter. There are many kinds of Internet access, and they are usually distinguished by how much bandwidth they provide and how much they cost, but the most important attribute is connectivity.

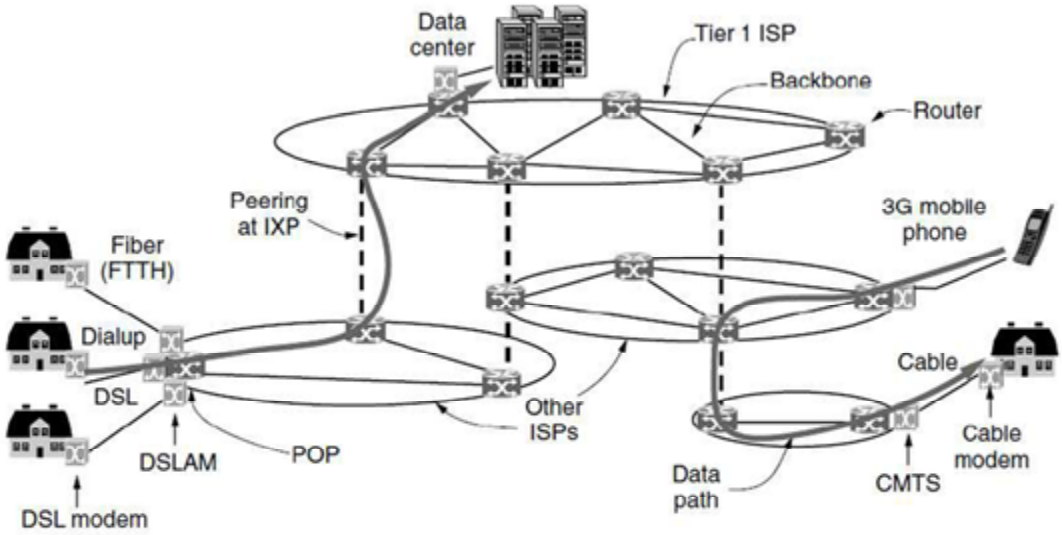


Figure 2.3 Overview of the Internet architecture

A common way to connect to an ISP is to use the phone line to your house, in which case your phone company is your ISP. DSL, short for Digital Subscriber Line, reuses the telephone line that connects to your house for digital data transmission. The computer is connected to a device called a DSL modem that converts between digital packets and analog signals that can pass unhindered over the telephone line. At the other end, a device called a DSLAM (Digital Subscriber Line Access Multiplexer) converts between signals and packets.

Several other popular ways to connect to an ISP are shown in Fig. 2.3. DSL is a higher-bandwidth way to use the local telephone line than to send bits over a traditional telephone call instead of a voice conversation. That is called dial-up and done with a different kind of modem at both ends. The word modem is short for "modulator demodulator" and refers to any device that converts between digital bits and analog signals. Another method is to send signals over the cable TV system. Like DSL, this is a way to reuse existing infrastructure, in this case otherwise unused cable TV channels. The device at the home end is called a cable modem and the device at the cable headend is called the CMTS (Cable Modem Termination System).

DSL and cable provide Internet access at rates from a small fraction of a megabit/sec to multiple megabit/sec, depending on the system. These rates are much greater than dial-up rates, which are limited to 56 kbps because of the narrow bandwidth used for voice calls.

Internet access at much greater than dial-up speeds is called broadband. The name refers to the broader bandwidth that is used for faster networks, rather than any particular speed.

The access methods mentioned so far are limited by the bandwidth of the "last mile" or last leg of transmission. By running optical fiber to residences, faster Internet access can be provided at rates on the order of 10 to 100 Mbps. This design is called FTTH (Fiber to the Home). For businesses in commercial areas, it may make sense to lease a high-speed transmission line from the offices to the nearest ISP. For example, in North America, a T3 line runs at roughly 45 Mbps.

Wireless is used for Internet access too. An example we will explore shortly is that of 3G mobile phone networks. They can provide data delivery at rates of 1 Mbps or higher to mobile phones and fixed subscribers in the coverage area.

We can now move packets between the home and the ISP. We call the location at which customer packets enter the ISP network for service the ISP's POP (Point of Presence). We will next explain how packets are moved between the POPs of different ISPs. From this point on, the system is fully digital and packet switched.

ISP networks may be regional, national, or international in scope. We have already seen that their architecture is made up of long-distance transmission lines that interconnect routers at POPs in the different cities that the ISPs serve. This equipment is called the backbone of the ISP. If a packet is destined for a host served directly by the ISP, that packet is routed over the backbone and delivered to the host. Otherwise, it must be handed over to another ISP.

ISPs connect their networks to exchange traffic at IXPs (Internet eXchange Points). The connected ISPs are said to peer with each other. There are many IXPs in cities around the world. They are drawn vertically in Fig. 2.3 because ISP networks overlap geographically. Basically, an IXP is a room full of routers, at least one per ISP. A LAN in the room connects all the routers, so packets can be forwarded from any ISP backbone to any other ISP backbone. IXPs can be large and independently owned facilities. One of the largest is the Amsterdam Internet Exchange, to which hundreds of ISPs connect and through which they exchange hundreds of gigabits/sec of traffic.

The peering that happens at IXPs depends on the business relationships between ISPs. There are many possible relationships. For example, a small ISP might pay a larger ISP for Internet connectivity to reach distant hosts, much as a customer purchases service from an Internet provider. In this case, the small ISP is said to pay for transit. Alternatively, two large ISPs might decide to exchangetraffic so that each ISP can deliver some traffic to the other ISP without having to pay for transit. One of the many paradoxes of the Internet is that ISPs who publicly compete with one another for customers often privately cooperate to do peering (Metz, 2001).

The path a packet takes through the Internet depends on the peering choices of the ISPs. If the ISP delivering a packet peers with the destination ISP, it might deliver the packet directly to its peer. Otherwise, it might route the packet to the nearest place at which it connects to a paid transit provider so that provider can deliver the packet. Two example paths across ISPs are drawn in Fig. 2.3. Often, the path a packet takes will not be the shortest path through the Internet.

At the top of the food chain are a small handful of companies, like AT&T and Sprint, that operate large international backbone networks with thousands of routers connected by high-bandwidth fiber optic links. These ISPs do not pay for transit. They are usually called tier 1 ISPs and are said to form the backbone of the Internet, since everyone else must connect to them to be able to reach the entire Internet.

Companies that provide lots of content, such as Google and Yahoo!, locate their computers in data centers that are well connected to the rest of the Internet. These data centers are designed for computers, not humans, and may be filled with rack upon rack of machines called a server farm. Colocation or hosting data centers let customers put equipment such as servers at ISP POPs so that short, fast connections can be made between the servers and the ISP backbones. The Internet hosting industry has become increasingly virtualized so that it is now common to rent a virtual machine that is run on a server farm instead of installing a physical computer. These data centers are so large (tens or hundreds of thousands of machines) that electricity is a major cost, so data centers are sometimes built in areas where electricity is cheap.

Also worth mentioning in passing is that some companies have interconnected all their existing internal networks, often using the same technology as the Internet. These intranets are typically accessible only on company premises or from company notebooks but otherwise work the same way as the Internet.

2.4 Third-Generation Mobile Phone Networks

People love to talk on the phone even more than they like to surf the Internet, and this has made the mobile phone network the most successful network in the world. The architecture of the mobile phone network has changed greatly over the past 40 years along with its tremendous growth. First-generation mobile phone systems transmitted voice calls as continuously varying (analog) signals rather than sequences of (digital) bits. AMPS (Advanced Mobile Phone System), which was deployed in the United States in 1982, was a widely used first generation system. Second-generation mobile phone systems switched to transmitting voice calls in digital form to increase capacity, improve security, and offer text messaging. GSM (Global System for Mobile communications), which was deployed starting in 1991 and has become the most widely used mobile phone system in the world, is a 2G system.

The third generation, or 3G, systems were initially deployed in 2001 and offer both digital voice and broadband digital data services. They also come with a lot of jargon and many different standards to choose from. 3G is loosely defined by the ITU (an international standards body) as providing rates of at least 2 Mbps for stationary or walking users and 384 kbps in a moving vehicle. UMTS (Universal Mobile Telecommunications System), also called WCDMA (Wideband Code Division Multiple Access), is the main 3G system that is being rapidly deployed worldwide. It can provide up to 14 Mbps on the downlink and almost 6 Mbps on the uplink. Future releases will use multiple antennas and radios to provide even greater speeds for users.

The scarce resource in 3G systems, as in 2G and 1G systems before them, is radio spectrum. Governments license the right to use parts of the spectrum to the mobile phone network operators, often using a spectrum auction in which network operators submit bids. Having a piece of licensed spectrum makes it easier to design and operate systems, since no one else is allowed transmit on that spectrum, but it often costs a serious amount of money. In the UK in 2000, for example, five 3G licenses were auctioned for a total of about $40 billion.

It is the scarcity of spectrum that led to the cellular network design shown in Fig. 2.4 that is now used for mobile phone networks. To manage the radio interference between users, the coverage area is divided into cells. Within a cell, users are assigned channels that do not interfere with each other and do not cause too much interference for adjacent cells. This allows for good reuse of the spectrum, or frequency reuse, in the neighboring cells, which increases the capacity of the network. In 1G systems, which carried each voice call on a specific frequency band, the frequencies were carefully chosen so that they did not conflict with neighboring cells. In this way, a given frequency might only be reused oncein several cells. Modern 3G systems allow each cell to use all frequencies, but in a way that results in a tolerable level of interference to the neighboring cells. There are variations on the cellular design, including the use of directional or sectored antennas on cell towers to further reduce interference, but the basic idea is the same.

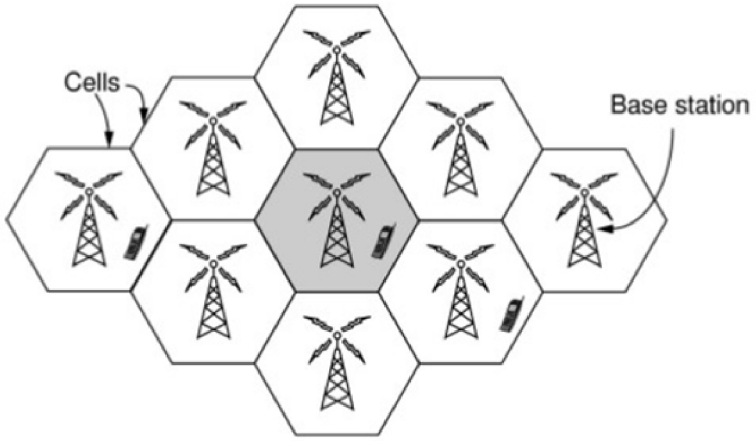


Figure 2.4 Cellular design of mobile phone networks

The architecture of the mobile phone network is very different than that of the Internet. It has several parts, as shown in the simplified version of the UMTS architecture in Fig. 2.5. First, there is the air interface. This term is a fancy name for the radio communication protocol that is used over the air between the mobile device (e.g., the cell phone) and the cellular base station. Advances in the air interface over the past decades have greatly increased wireless data rates. The UMTS air interface is based on a technique called Code Division Multiple Access (CDMA).

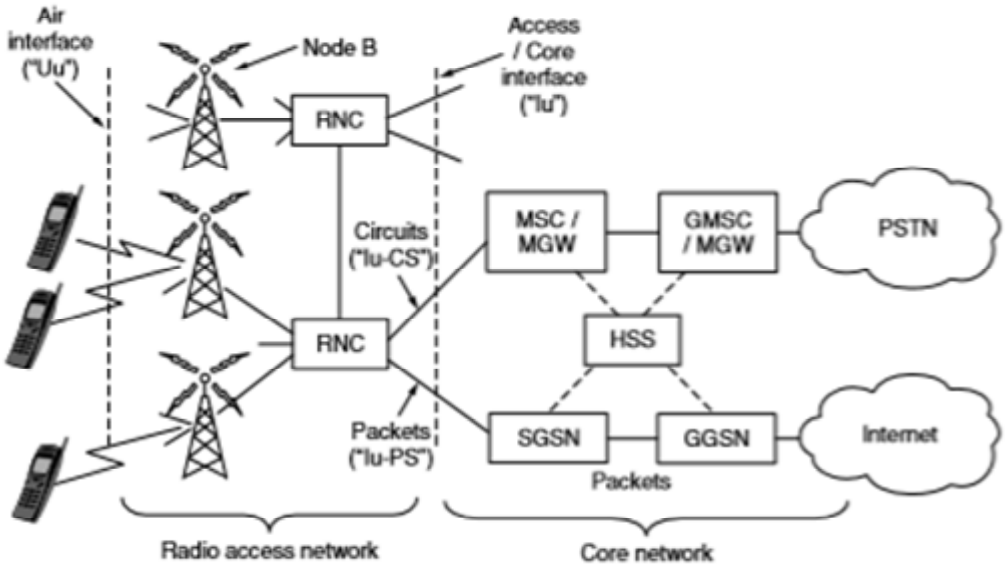


Figure 2.5 Architecture of the UMTS 3G mobile phone network

The cellular base station together with its controller forms the radio access network. This part is the wireless side of the mobile phone network. The controller node or RNC (Radio Network Controller) controls how the spectrum is used. The base station implements the air interface. It is called Node B, a temporary label that stuck.

The rest of the mobile phone network carries the traffic for the radio access network. It is called the core network. The UMTS core network evolved from the core network used for the 2G GSM system that came before it. However, something surprising is happening in the UMTS core network.

Since the beginning of networking, a war has been going on between the people who support packet networks (i.e., connectionless subnets) and the people who support circuit networks (i.e., connection-oriented subnets). The main proponents of packets come from the Internet community. In a connectionless design, every packet is routed independently of every other packet. As a consequence, if some routers go down during a session, no harm will be done as long as the system can dynamically reconfigure itself so that subsequent packets can find some route to the destination, even if it is different from that which previous packets used.

The circuit camp comes from the world of telephone companies. In the telephone system, a caller must dial the called party's number and wait for a connection before talking or sending data. This connection setup establishes a route through the telephone system that is maintained until the call is terminated. All words or packets follow the same route. If a line or switch on the path goes down, the call is aborted, making it less fault tolerant than a connectionless design.

The advantage of circuits is that they can support quality of service more easily. By setting up a connection in advance, the subnet can reserve resources such as link bandwidth, switch buffer space, and CPU. If an attempt is made to set up a call and insufficient resources are available, the call is rejected and the caller gets a kind of busy signal. In this way, once a connection has been set up, the connection will get good service.

With a connectionless network, if too many packets arrive at the same router at the same moment, the router will choke and probably lose packets. The sender will eventually notice this and resend them, but the quality of service will be jerky and unsuitable for audio or video unless the network is lightly loaded. Needless to say, providing adequate audio quality is something telephone companies care about very much, hence their preference for connections.

In Fig. 2.5, both the packet and circuit switched equipment are present in the core network. This shows the mobile phone network in transition, with mobile phone companies able to implement one or sometimes both ofthe alternatives. Older mobile phone networks used a circuit-switched core in the style of the traditional phone network to carry voice calls. This legacy is seen in the UMTS network with the MSC (Mobile Switching Center), GMSC (Gateway Mobile Switching Center), and MGW (Media Gateway) elements that set up connections over a circuit-switched core network such as the PSTN (Public Switched Telephone Network).

Data services have become a much more important part of the mobile phone network than they used to be, starting with text messaging and early packet data services such as GPRS (General Packet Radio Service) in the GSM system. These older data services ran at tens of kbps, but users wanted more. Newer mobile phone networks carry packet data at rates of multiple Mbps. For comparison, a voice call is carried at a rate of 64 kbps, typically 3-4x less with compression.

To carry all this data, the UMTS core network nodes connect directly to a packet-switched network. The SGSN (Serving GPRS Support Node) and the GGSN (Gateway GPRS Support Node) deliver data packets to and from mobiles and interface to external packet networks such as the Internet.

This transition is set to continue in the mobile phone networks that are now being planned and deployed. Internet protocols are even used on mobiles to set up connections for voice calls over a packet data network, in the manner of voiceover-IP. IP and packets are used all the way from the radio access through to the core network. Of course, the way that IP networks are designed is also changing to support better quality of service. If it did not, then problems with chopped-up audio and jerky video would not impress paying customers.

Another difference between mobile phone networks and the traditional Internet is mobility. When a user moves out of the range of one cellular base station and into the range of another one, the flow of data must be re-routed from the old to the new cell base station. This technique is known as handover or handoff, and it is illustrated in Fig. 2.6.

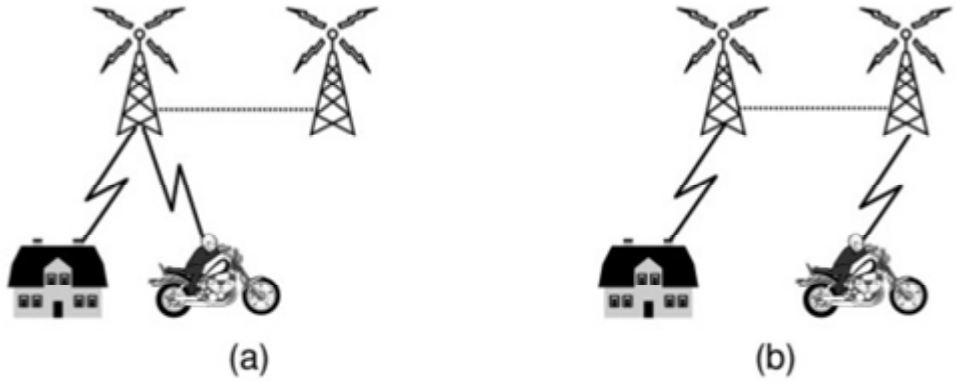


Figure 2.6 Mobile phone handover (a) before, (b) after

Either the mobile device or the base station may request a handover when the quality of the signal drops. In some cell networks, usually those based on CDMA technology, it is possible to connect to the new base station before disconnecting from the old base station. This improves the connection quality for the mobile because there is no break in service; the mobile is actually

connected to two base stations for a short while. This way of doing a handover is called a soft handover to distinguish it from a hard handover, in which the mobile disconnects from the old base station before connecting to the new one.

A related issue is how to find a mobile in the first place when there is an incoming call. Each mobile phone network has aHSS (Home Subscriber Server) in the core network that knows the location of each subscriber, as well as other profile information that is used for authentication and authorization. In this way, each mobile can be found by contacting the HSS.

A final area to discuss is security. Historically, phone companies have taken security much more seriously than Internet companies for a long time because of the need to bill for service and avoid (payment) fraud. Unfortunately that is not saying much. Nevertheless, in the evolution from 1G through 3G technologies, mobile phone companies have been able to roll out some basic security mechanisms for mobiles.

Starting with the 2G GSM system, the mobile phone was divided into a handset and a removable chip containing the subscriber's identity and account information. The chip is informally called a SIM card, short for Subscriber Identity Module. SIM cards can be switched to different handsets to activate them, and they provide a basis for security. When GSM customers travel to other countries on vacation or business, they often bring their handsets but buy a new SIM card for few dollars upon arrival in order to make local calls with no roaming charges.

To reduce fraud, information on SIM cards is also used by the mobile phone network to authenticate subscribers and check that they are allowed to use the network. With UMTS, the mobile also uses the information on the SIM card to check that it is talking to a legitimate network.

Another aspect of security is privacy. Wireless signals are broadcast to all nearby receivers, so to make it difficult to eavesdrop on conversations, cryptographic keys on the SIM card are used to encrypt transmissions. This approach provides much better privacy than in 1G systems, which were easily tapped, but is not a panacea due to weaknesses in the encryption schemes.

Mobile phone networks are destined to play a central role in future networks. They are now more about mobile broadband applications than voice calls, and this has major implications

for the air interfaces, core network architecture, and security of future networks. 4G technologies that are faster and better are on the drawing board under the name of LTE (Long Term Evolution), even as 3G design and deployment continues. Other wireless technologies also offer broadband Internet access to fixed and mobile clients, notably 802.16 networks under the common name of WiMAX. It is entirely possible that LTE and WiMAX are on a collision course with each other and it is hard to predict what will happen to them.

2.5 Wireless LANs: 802.11

Almost as soon as laptop computers appeared, many people had a dream of walking into an office and magically having their laptop computer be connected to the Internet. Consequently, various groups began working on ways to accomplish this goal. The most practical approach is to equip both the office and the laptop computers with short-range radio transmitters and receivers to allow them to talk.

Work in this field rapidly led to wireless LANs being marketed by a variety of companies. The trouble was that no two of them were compatible. The proliferation of standards meant that a computer equipped with a brand X radio would not work in a room equipped with a brand Y base station. In the mid 1990s, the industry decided that a wireless LAN standard might be a good idea, so the IEEE committee that had standardized wired LANs was given the task of drawing up a wireless LAN standard.

The first decision was the easiest: what to call it. All the other LAN standards had numbers like 802.1, 802.2, and 802.3, up to 802.10, so the wireless LAN standard was dubbed 802.11. A common slang name for it is WiFi but it is an important standard and deserves respect, so we will call it by its proper name, 802.11.

The rest was harder. The first problem was to find a suitable frequency band that was available, preferably worldwide. The approach taken was the opposite of that used in mobile phone networks. Instead of expensive, licensed spectrum, 802.11 systems operate in unlicensed bands such as the ISM (Industrial, Scientific, and Medical) bands defined by ITU-R (e.g., 902-928 MHz, 2.4-2.5 GHz, 5.725-5.825 GHz). All devices are allowed to use this spectrum provided that they limit their transmit power to let different devices coexist. Of course, this means that

802.11 radios may find themselves competing with cordless phones, garage door openers, and microwave ovens.

802.11 networks are made up of clients, such as laptops and mobile phones, and infrastructure called APs (access points) that is installed in buildings. Access points are sometimes called base stations. The access points connect to the wired network, and all communication between clients goes through an access point. It is also possible for clients that are in radio range to talk directly, such as two computers in an office without an access point. This arrangement is called an ad hoc network. It is used much less often than the access point mode. Both modes are shown in Fig. 2.7.

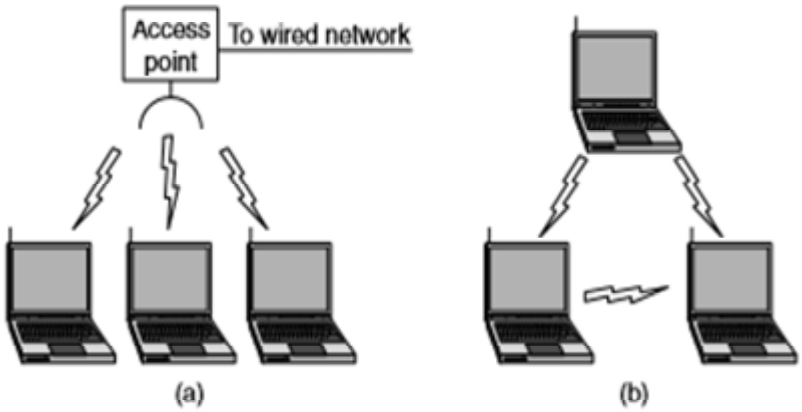


Figure 2.7 (a) Wireless network with an access point. (b) Ad hoc network.

802.11 transmission is complicated by wireless conditions that vary with even small changes in the environment. At the frequencies used for 802.11, radio signals can be reflected off solid objects so that multiple echoes of a transmission may reach a receiver along different paths. The echoes can cancel or reinforce each other, causing the received signal to fluctuate greatly. This phenomenon is called multipath fading, and it is shown in Fig. 2.8.

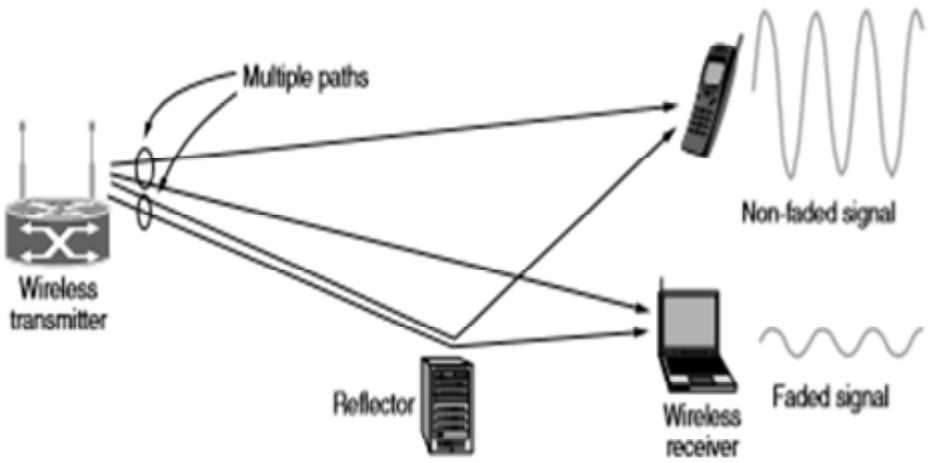


Figure 2.8 : Multipath fading

The key idea for overcoming variable wireless conditions is path diversity, or the sending of information along multiple, independent paths. In this way, theinformation is likely to be received even if one of the paths happens to be poor due to a fade. These independent paths are typically built into the digital modulation scheme at the physical layer. Options include using different frequencies across the allowed band, following different spatial paths between different pairs of antennas, or repeating bits over different periods of time.

Since wireless is inherently a broadcast medium, 802.11 radios also have to deal with the problem that multiple transmissions that are sent at the same time will collide, which may interfere with reception. To handle this problem, 802.11 uses a CSMA (Carrier Sense Multiple Access) scheme that draws on ideas from classic wired Ethernet, which, ironically, drew from an early wireless network developed in Hawaii and called ALOHA. Computers wait for a short random interval before transmitting, and defer their transmissions if they hear that someone else is already transmitting. This scheme makes it less likely that two computers will send at the same time. It does not work as well as in the case of wired networks, though. To see why, examine Fig. 2.9. Suppose that computer A is transmitting to computer B, but the radio range of A's transmitter is too short to reach computer C. If C wants to transmit to B it can listen before starting, but the fact that it does not hear anything does not mean that its transmission will succeed. The inability of C to hear A before starting causes some collisions to occur. After any collision, the sender then waits another, longer, random delay and retransmits the packet. Despite this and some other issues, the scheme works well enough in practice.

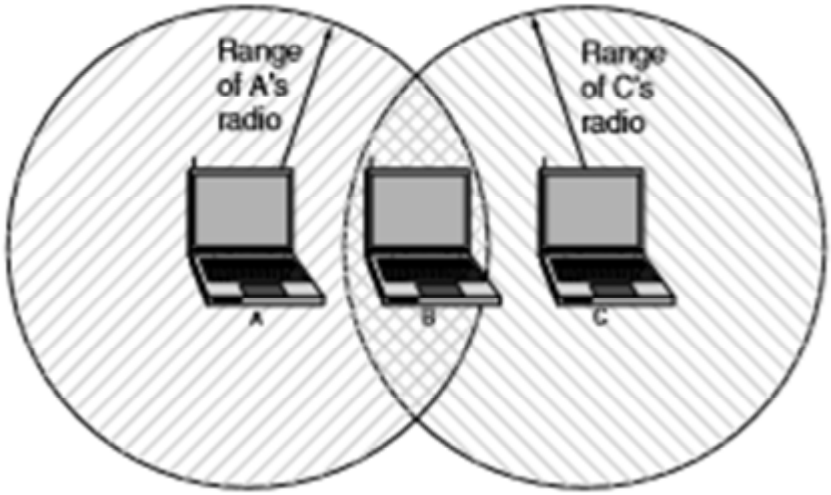


Figure 2.9 : The range of a single radio may not cover the entire system

Another problem is that of mobility. If a mobile client is moved away from the access point it is using and into the range of a different access point, some way of handing it off is needed. The solution is that an 802.11 network can consist of multiple cells, each with its own access point, and a distribution system that connects the cells. The distribution system is often switched Ethernet, but it can use any technology. As the clients move, they may find another access point with a better signal than the one they are currently using and change their association. From the outside, the entire system looks like a single wired LAN.

That said, mobility in 802.11 has been of limited value so far compared to mobility in the mobile phone network. Typically, 802.11 is used by nomadic clients that go from one fixed location to another, rather than being used on-the-go. Mobility is not really needed for nomadic usage. Even when 802.11 mobility is used, it extends over a single 802.11 network, which might cover at most a large building. Future schemes will need to provide mobility across different networks and across different technologies (e.g., 802.21).

Finally, there is the problem of security. Since wireless transmissions are broadcast, it is easy for nearby computers to receive packets of information that were not intended for them. To prevent this, the 802.11 standard included an encryption scheme known as WEP (Wired Equivalent Privacy). The idea was to make wireless security like that of wired security. It is a good idea, but unfortunately the scheme was flawed and soon broken (Borisov et al., 2001). It

has since been replaced with newer schemes that have different cryptographic details in the 802.11i standard, also called WiFi Protected Access, initially called WPA but now replaced by WPA2.

802.11 has caused a revolution in wireless networking that is set to continue. Beyond buildings, it is starting to be installed in trains, planes, boats, and automobiles so that people can surf the Internet wherever they go. Mobile phones and all manner of consumer electronics, from game consoles to digital cameras, can communicate with it.

2.6 RFID and Sensor Networks

The networks we have studied so far are made up of computing devices that are easy to recognize, from computers to mobile phones. With Radio Frequency IDentification (RFID), everyday objects can also be part of a computer network.

An RFID tag looks like a postage stamp-sized sticker that can be affixed to (or embedded in) an object so that it can be tracked. The object might be a cow, a passport, a book or a shipping pallet. The tag consists of a small microchip with a unique identifier and an antenna that receives radio transmissions. RFID readers installed at tracking points find tags when they come into range and interrogate them for their information. Applications include checking identities, managing the supply chain, timing races, and replacing barcodes.

There are many kinds of RFID, each with different properties, but perhaps the most fascinating aspect of RFID technology is that most RFID tags have neither an electric plug nor a battery. Instead, all of the energy needed to operate them is supplied in the form of radio waves by RFID readers. This technology is called passive RFID to distinguish it from the (less common) active RFID in which there is a power source on the tag.

One common form of RFID is UHF RFID (Ultra-High Frequency RFID). It is used on shipping pallets and some drivers licenses. Readers send signals inthe 902-928 MHz band in the United States. Tags communicate at distances of several meters by changing the way they reflect the reader signals; the reader is able to pick up these reflections. This way of operating is called backscatter.

Another popular kind of RFID is HF RFID (High Frequency RFID). It operates at 13.56 MHz and is likely to be in your passport, credit cards, books, and noncontact payment systems. HF RFID has a short range, typically a meter or less, because the physical mechanism is based on induction rather than backscatter. There are also other forms of RFID using other frequencies, such as LF RFID (Low Frequency RFID), which was developed before HF RFID and used for animal tracking. It is the kind of RFID likely to be in your cat.

RFID readers must somehow solve the problem of dealing with multiple tags within reading range. This means that a tag cannot simply respond when it hears a reader, or the signals from multiple tags may collide. The solution is similar to the approach taken in 802.11: tags wait for a short random interval before responding with their identification, which allows the reader to narrow down individual tags and interrogate them further.

Security is another problem. The ability of RFID readers to easily track an object, and hence the person who uses it, can be an invasion of privacy. Unfortunately, it is difficult to secure RFID tags because they lack the computation and communication power to run strong cryptographic algorithms. Instead, weak measures like passwords (which can easily be cracked) are used. If an identity card can be remotely read by an official at a border, what is to stop the same card from being tracked by other people without your knowledge? Not much.

RFID tags started as identification chips, but are rapidly turning into full-fledged computers. For example, many tags have memory that can be updated and later queried, so that information about what has happened to the tagged object can be stored with it. Rieback et al. (2006) demonstrated that this means that all of the usual problems of computer malware apply, only now your cat or your passport might be used to spread an RFID virus.

A step up in capability from RFID is the sensor network. Sensor networks are deployed to monitor aspects of the physical world. So far, they have mostly been used for scientific experimentation, such as monitoring bird habitats, volcanic activity, and zebra migration, but business applications including healthcare,monitoring equipment for vibration, and tracking of frozen, refrigerated, or otherwise perishable goods cannot be too far behind.

Sensor nodes are small computers, often the size of a key fob, that have temperature, vibration, and other sensors. Many nodes are placed in the environment that is to be monitored.

Typically, they have batteries, though they may scavenge energy from vibrations or the sun. As with RFID, having enough energy is a key challenge, and the nodes must communicate carefully to be able to deliver their sensor information to an external collection point. A common strategy is for the nodes to self-organize to relay messages for each other, as shown in Fig. 2.10. This design is called a multihop network.

Figure 2.10 Multihop topology of a sensor network

RFID and sensor networks are likely to become much more capable and pervasive in the future. Researchers have already combined the best of both technologies by prototyping programmable RFID tags with light, movement, and other sensors (Sample et al., 2008).

2.7 Summary

Networks provide various services to their users. These services can range from connectionless best-efforts packet delivery to connection-oriented guaranteed delivery. In some networks, connectionless service is provided in one layer and connection-oriented service is provided in the layer above it.

Well-known networks include the Internet, the 3G mobile telephone network, and 802.11 LANs. The Internet evolved from the ARPANET, to which other networks were added to form an internetwork. The present-day Internet is actually a collection of many thousands of networks that use the TCP/IP protocol stack. The 3G mobile telephone network provides wireless and mobile access to the Internet at speeds of multiple Mbps, and, of course, carries voice calls as

well. Wireless LANs based on the IEEE 802.11 standard are deployed in many homes and cafes and can provide connectivity at rates in excess of 100 Mbps. New kinds of networks are emerging too, such as embedded sensor networks and networks based on RFID technology.

2.8 Model Questions

1. Write short notes on Wireless LAN.
2. Describe the structure of the Telephone system.
3. Discuss the properties of Ethernet.
4. Explain in detail about the Internet architecture.
5. Discuss about the 3G mobile phone networks in detail.

LESSON-3
REFERENCE MODELS

Structure

3.1 Introduction

In this lesson, we will discuss two important network architectures: the OSI reference model and the TCP/IP reference model. Although the protocols associated with the OSI model are not used any more, the model itself is actually quite general and still valid, and the features discussed at each layer are still very important. The TCP/IP model has the opposite properties: the model itself is not of much use but the protocols are widely used. For this reason we will look at both of them in detail.

3.2 Objectives

- To explain the two important reference models.
- To discuss the differences between the OSI and TCP/IP reference models.

3.3 The OSI Reference Model

The OSI model (minus the physical medium) is shown in Fig. 3.1. This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann,

1983). It was revised in 1995 (Day, 1995). The model is called the ISO OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems-that is, systems that are open for communication with other systems.

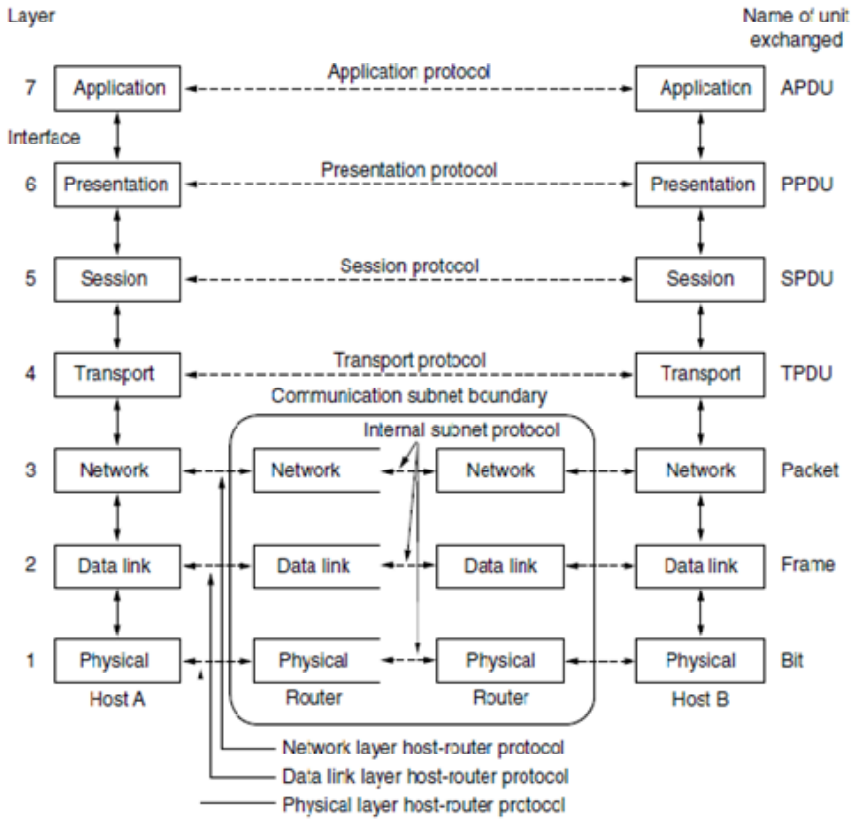


Figure 3.1 The OSI reference model

The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.

5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

Below we will discuss each layer of the model in turn, starting at the bottom layer. Note that the OSI model itself is not a network architecture because it does not specify the exact services and protocols to be used in each layer. It just tells what each layer should do. However, ISO has also produced standards for all the layers, although these are not part of the reference model itself. Each one has been published as a separate international standard.

3.3.1 The Physical Layer

The physical layer is concerned with transmitting raw bits over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit it is received by the other side as a 1 bit, not as a 0 bit. Typical questions here are what electrical signals should be used to represent a 1 and a 0, how many nanoseconds a bit lasts, whether transmission may proceed simultaneously in both directions, how the initial connection is established, how it is torn down when both sides are finished, how many pins the network connector has, and what each pin is used for. These design issues largely deal with mechanical, electrical, and timing interfaces, as well as the physical transmission medium, which lies below the physical layer.

3.3.2 The Data Link Layer

The main task of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors. It does so by masking the real errors so the network layer does not see them. It accomplishes this task by having the sender break up the input data into data frames (typically a few hundred or a few thousand bytes) and transmit the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an acknowledgement frame.

Another issue that arises in the data link layer (and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism may be needed to let the transmitter know when the receiver can accept more data.

Broadcast networks have an additional issue in the data link layer: how to control access to the shared channel. A special sublayer of the data link layer, the medium access control sublayer, deals with this problem.

3.3.3 The Network Layer

The network layer controls the operation of the subnet. A key design issue is determining how packets are routed from source to destination. Routes can be based on static tables that are "wired into" the network and rarely changed, or more often they can be updated automatically to avoid failed components. They can also be determined at the start of each conversation, for example, a terminal session, such as a login to a remote machine. Finally, they can be highly dynamic, being determined anew for each packet to reflect the current network load. If too many packets are present in the subnet at the same time, they will get in one another's way, forming bottlenecks. Handling congestion is also a responsibility of the network layer, in conjunction with higher layers that adapt the loadthey place on the network. More generally, the quality of service provided (delay, transit time, jitter, etc.) is also a network layer issue.

When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from that used by the first one. The second one may not accept the packet at all because it is too large. The protocols may differ, and so on. It is up to the network layer to overcome all these problems to allow heterogeneous networks to be interconnected.

In broadcast networks, the routing problem is simple, so the network layer is often thin or even nonexistent.

3.3.4 The Transport Layer

The basic function of the transport layer is to accept data from above it, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology over the course of time.

The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service exist, such as the transporting of isolated messages with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations. The type of service is determined when the connection is established.

The transport layer is a true end-to-end layer; it carries data all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers, each protocols is between a machine and its immediate neighbors, and not between the ultimate source and destination machines, which may be separated by many routers. The difference between layers 1 through 3, which are chained, and layers 4 through 7, which are end-to-end, is illustrated in Fig. 3.1.

3.3.5 The Session Layer

The session layer allows users on different machines to establish sessions between them. Sessions offer various services, including dialog control (keeping track of whose turn it is to transmit), token management (preventing two parties from attempting the same critical operation simultaneously), and synchronization(checkpointing long transmissions to allow them to pick up from where they left off in the event of a crash and subsequent recovery).

3.3.6 The Presentation Layer

Unlike the lower layers, which are mostly concerned with moving bits around, the presentation layer is concerned with the syntax and semantics of the information transmitted. In order to make it possible for computers with different internal data representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used "on the wire." The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records) to be defined and exchanged.

3.3.7 The Application Layer

The application layer contains a variety of protocols that are commonly needed by users. One widely used application protocol is HTTP (HyperText Transfer Protocol), which is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server hosting the page using HTTP. The server then sends the page back. Other application protocols are used for file transfer, electronic mail, and network news.

3.4 The TCP/IP Reference Model

Let us now turn from the OSI reference model to the reference model used in the grandparent of all wide area computer networks, the ARPANET, and its successor, the worldwide Internet. Although we will give a brief history of the ARPANET later, it is useful to mention a few key aspects of it now. The ARPANET was a research network sponsored by the DoD (U.S. Department of Defense). It eventually connected hundreds of universities and government installations, using leased telephone lines. When satellite and radio networks were added later, the existing protocols had trouble interworking with them, so a new reference architecture was needed. Thus, from nearly the beginning, the ability to connect multiple networks in a seamless way was one of the major design goals. This architecture later became known as the TCP/IP Reference Model, after its two primary protocols. It was first described by Cerf and Kahn (1974), and later refined and defined as a standard in the Internet community (Braden, 1989). The design philosophy behind the model is discussed by Clark (1988).

Given the DoD's worry that some of its precious hosts, routers, and internetwork gateways might get blown to pieces at a moment's notice by an attack from the Soviet Union, another major goal was that the network be able to survive loss of subnet hardware, without existing conversations being broken off. In otherwords, the DoD wanted connections to remain intact as long as the source and destination machines were functioning, even if some of the machines or transmission lines in between were suddenly put out of operation. Furthermore, since applications with divergent requirements were envisioned, ranging from transferring files to real-time speech transmission, a flexible architecture was needed.

3.4.1 The Link Layer

All these requirements led to the choice of a packet-switching network based on a connectionless layer that runs across different networks. The lowest layer in the model, the link layer describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer. It is not really a layer at all, in the normal sense of the term, but rather an interface between hosts and transmission links. Early material on the TCP/IP model has little to say about it.

3.4.2 The Internet Layer

The internet layer is the linchpin that holds the whole architecture together. It is shown in Fig. 3.2 as corresponding roughly to the OSI network layer. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination (potentially on a different network). They may even arrive in a completely different order than they were sent, in which case it is the job of higher layers to rearrange them, if in-order delivery is desired. Note that "internet" is used here in a generic sense, even though this layer is present in the Internet.

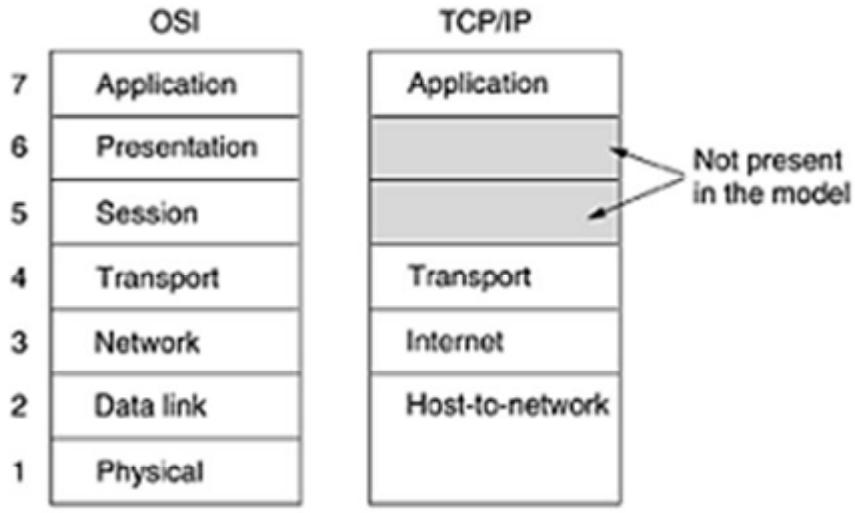


Figure 3.2 The TCP/IP reference model

The analogy here is with the (snail) mail system. A person can drop a sequence of international letters into a mailbox in one country, and with a little luck,most of them will be delivered to the correct address in the destination country. The letters will probably travel through

one or more international mail gateways along the way, but this is transparent to the users. Furthermore, that each country (i.e., each network) has its own stamps, preferred envelope sizes, and delivery rules is hidden from the users.

The internet layer defines an official packet format and protocol called IP (Internet Protocol), plus a companion protocol called ICMP (Internet Control Message Protocol) that helps it function. The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing is clearly a major issue here, as is congestion.

3.4.3 The Transport Layer

The layer above the internet layer in the TCP/IP model is now usually called the transport layer. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer. Two end-to-end transport protocols have been defined here. The first one, TCP (Transmission Control Protocol), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the internet. It segments the incoming byte stream into discrete messages and passes each one on to the internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.

The second protocol in this layer, UDP (User Datagram Protocol), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video. The relation of IP, TCP, and UDP is shown in Fig. 2.3. Since the model was developed, IP has been implemented on many other networks.

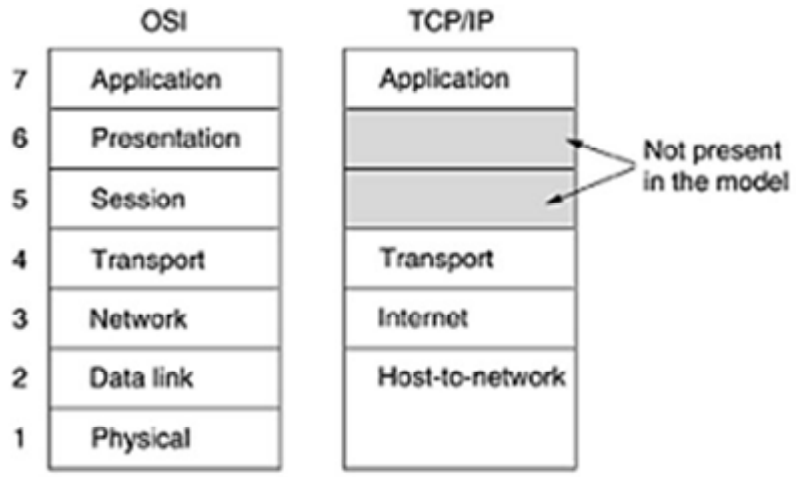


Figure 3.3 The TCP/IP model with some protocols

3.4.4 The Application Layer

The TCP/IP model does not have session or presentation layers. No need for them was perceived. Instead, applications simply include any session and presentation functions that they require. Experience with the OSI model has proven this view correct: these layers are of little use to most applications.

On top of the transport layer is the application layer. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP, Simple Mail Transfer Protocol). Many other protocols have been added to these over the years. Some important ones that we will study, include the Domain Name System (DNS), for mapping host names onto their network addresses, HTTP (HyperText Transfer Protocol), the protocol for fetching pages on the World Wide Web, and RTP (Real-Time Transport Protocol), the protocol for delivering real-time media such as voice or movies.

3.4.5 The Hybrid Reference Model

As mentioned earlier, the strength of the OSI reference model is the model itself (minus the presentation and session layers), which has proven to be exceptionally useful for discussing computer networks. In contrast, the strength of the TCP/IP reference model is the protocols,

which have been widely used for many years. Since computer scientists like to have their cake and eat it, too, we will use the hybrid model of Fig. 3.4 as the framework.

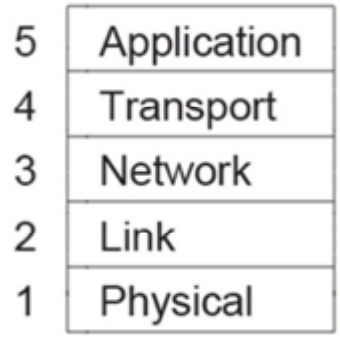


Figure 3.4 The hybrid reference model

This model has five layers, running from the physical layer up through the link, network and transport layers to the application layer. The physical layer specifies how to transmit bits across different kinds of media as electrical (or other analog) signals. The link layer is concerned with how to send finite-length messages between directly connected computers with specified levels of reliability. Ethernet and 802.11 are examples of link layer protocols.

The network layer deals with how to combine multiple links into networks, and networks of networks, into internetworks so that we can send packets between distant computers. This includes the task of finding the path along which to send the packets. IP is the main example protocol we will study for this layer. The transport layer strengthens the delivery guarantees of the Network layer, usually with increased reliability, and provide delivery abstractions, such as a reliable byte stream, that match the needs of different applications. TCP is an important example of a transport layer protocol.

Finally, the application layer contains programs that make use of the network. Many, but not all, networked applications have user interfaces, such as a Web browser. Our concern, however, is with the portion of the program that uses the network. This is the HTTP protocol in the case of the Web browser. There are also important support programs in the application layer, such as the DNS, that are used by many applications.

3.5 A Comparison - OSI and TCP/IP Reference Models

The OSI and TCP/IP reference models have much in common. Both are based on the concept of a stack of independent protocols. Also, the functionality of the layers is roughly similar. For example, in both models the layers up through and including the transport layer are there to provide an end-to-end, network-independent transport service to processes wishing to communicate. These layers form the transport provider. Again in both models, the layers above transport are application-oriented users of the transport service.

Despite these fundamental similarities, the two models also have many differences. In this section we will focus on the key differences between the two reference models. It is important to note that we are comparing the reference models here, not the corresponding protocol stacks. Three concepts are central to the OSI model:

1. Services.
2. Interfaces.
3. Protocols.

Probably the biggest contribution of the OSI model is that it makes the distinction between these three concepts explicit. Each layer performs some services for the layer above it. The service definition tells what the layer does, not how entities above it access it or how the layer works. It defines the layer's semantics.

A layer's interface tells the processes above it how to access it. It specifies what the parameters are and what results to expect. It, too, says nothing about how the layer works inside.

Finally, the peer protocols used in a layer are the layer's own business. It can use any protocols it wants to, as long as it gets the job done (i.e., provides the offered services). It can also change them at will without affecting software in higher layers.

These ideas fit very nicely with modern ideas about object-oriented programming. An object, like a layer, has a set of methods (operations) that processes outside the object can invoke. The semantics of these methods define the set of services that the object offers. The

methods' parameters and results form the object's interface. The code internal to the object is its protocol and is not visible or of any concern outside the object.

The TCP/IP model did not originally clearly distinguish between services, interfaces, and protocols, although people have tried to retrofit it after the fact to make it more OSI-like. For example, the only real services offered by the internet layer are SEND IP PACKET and RECEIVE IP PACKET. As a consequence, the protocols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes. Being able to make such changes transparently is one of the main purposes of having layered protocols in the first place.

The OSI reference model was devised before the corresponding protocols were invented. This ordering meant that the model was not biased toward one particular set of protocols, a fact that made it quite general. The downside of this ordering was that the designers did not have much experience with the subject and did not have a good idea of which functionality to put in which layer.

For example, the data link layer originally dealt only with point-to-point networks. When broadcast networks came around, a new sublayer had to be hacked into the model. Furthermore, when people started to build real networks using the OSI model and existing protocols, it was discovered that these networks did not match the required service specifications (wonder of wonders), so convergence sublayers had to be grafted onto the model to provide a place for papering over the differences. Finally, the committee originally expected that each country would have one network, run by the government and using the OSI protocols, so no thought was given to internetworking. To make a long story short, things did not turn out that way.

With TCP/IP the reverse was true: the protocols came first, and the model was really just a description of the existing protocols. There was no problem with the protocols fitting the model. They fit perfectly. The only trouble was that the model did not fit any other protocol stacks. Consequently, it was not especially useful for describing other, non-TCP/IP networks.

Turning from philosophical matters to more specific ones, an obvious difference between the two models is the number of layers: the OSI model has seven layers and the TCP/IP model

has four. Both have (inter)network, transport, and application layers, but the other layers are different.

Another difference is in the area of connectionless versus connection-oriented communication. The OSI model supports both connectionless and connection-oriented communication in the network layer, but only connection-oriented communication in the transport layer, where it counts (because the transport service is visible to the users). The TCP/IP model supports only one mode in the network layer (connectionless) but both in the transport layer, giving the users a choice. This choice is especially important for simple request-response protocols.

3.6 Summary

Network software is built around protocols, which are rules by which processes communicate. Most networks support protocol hierarchies, with each layer providing services to the layer above it and insulating them from the details of the protocols used in the lower layers. Protocol stacks are typically based either on the OSI model or on the TCP/IP model. Both have link, network, transport, and application layers, but they differ on the other layers. Design issues include reliability, resource allocation, growth, security, and more.

3.7 Model Questions

1. Explain the OSI reference model.
2. Write in detail about the TCP/IP reference model.
3. Explain the steps in establishing and releasing a Connection in Transport layer.
4. Describe OSI and TCP/IP Reference models and compare them.

LESSON - 4
PHYSICAL LAYER

Structure

4.1 Introduction

In this lesson, we will look at the lowest layer in our protocol model, the physical layer. It defines the electrical, timing and other interfaces by which bits are sent as signals over channels. The physical layer is the foundation on which the network is built. The properties of different kinds of physical channels determine the performance (e.g., throughput, latency, and error rate).

We will begin with transmission media. There are three kinds of transmission media: guided (copper wire and fiber optics), wireless (terrestrial radio), and satellite. Each of these technologies has different properties that affect the design and performance of the networks that use them. This material will provide background information on the key transmission technologies used in modern networks.

Next comes digital modulation, which is all about how analog signals are converted into digital bits and back again. After that we will look at multiplexing schemes, exploring how multiple conversations can be put on the same transmission medium at the same time without interfering with one another.

Finally, we will look at three examples of communication systems used in practice for wide area computer networks: the (fixed) telephone system, the mobile phone system, and the

cable television system. Each of these is important in practice, so we will devote a fair amount of space to each one.

4.2 Objectives

- To explain the basics of physical layer.
- To discuss about the guided transmission media and its types.

4.3 GUIDED TRANSMISSION MEDIA

The purpose of the physical layer is to transport bits from one machine to another.Various physical media can be used for the actual transmission. Each one has its own niche in terms of bandwidth, delay, cost, and ease of installation and maintenance. Media are roughly grouped into guided media, such as copper wire and fiber optics, and unguided media, such as terrestrial wireless, satellite, and lasers through the air. We will look at guided media in this section.

4.3.1 Magnetic Media

One of the most common ways to transport data from one computer to another is to write them onto magnetic tape or removable media (e.g., recordable DVDs),physically transport the tape or disks to the destination machine, and read them back in again. Although this method is not as sophisticated as using a geosynchronous communication satellite, it is often more cost effective, especially for applications in which high bandwidth or cost per bit transported is the key factor.

A simple calculation will make this point clear. An industry-standard Ultriumtape can hold 800 gigabytes. A box 60 × 60 × 60 cm can hold about 1000 of these tapes, for a total capacity of 800 terabytes, or 6400 terabits (6.4 petabits). A box of tapes can be delivered anywhere in the United States in 24 hours by Federal Express and other companies. The effective bandwidth of this transmission is6400 terabits/86,400 sec, or a bit over 70 Gbps. If the destination is only an hour away by road, the bandwidth is increased to over 1700 Gbps. No computer network can even approach this. Of course, networks are getting faster, but tape densities are increasing, too.

If we now look at cost, we get a similar picture. The cost of an Ultrium tape is around $40 when bought in bulk. A tape can be reused at least 10 times, so the tape cost is may be $4000 per box per usage. Add to this another $1000 for shipping(probably much less), and we have a cost of roughly $5000 to ship 800 TB.This amounts to shipping a gigabyte for a little over half a cent. No network can beat that.

4.3.2 Twisted Pairs

Although the bandwidth characteristics of magnetic tape are excellent, the delay characteristics are poor. Transmission time is measured in minutes or hours,not milliseconds. For many applications an online connection is needed. One of the oldest and still most common transmission media is twisted pair. A twisted pair consists of two insulated copper wires, typically about 1 mm thick. The wires are twisted together in a helical form, just like a DNA molecule. Twisting is done because two parallel wires constitute a fine antenna. When the wires are twisted, the waves from different twists cancel out, so the wire radiates less effectively. A signal is usually carried as the difference in voltage between the two wires in the pair. This provides better immunity to external noise because the noise tends to affect both wires the same, leaving the differential unchanged.

The most common application of the twisted pair is the telephone system.Nearly all telephones are connected to the telephone company (telco) office by a twisted pair. Both telephone calls and ADSL Internet access run over these lines.Twisted pairs can run several kilometers without amplification, but for longer distances the signal becomes too attenuated and repeaters are needed. When many twisted pairs run in parallel for a substantial distance, such as all the wires coming from an apartment building to the telephone company office, they are bundled together and encased in a protective sheath. The pairs in these bundles would interfere with one another if it were not for the twisting. In parts of the world where telephone lines run on poles above ground, it is common to see bundles several centimeters in diameter.

Twisted pairs can be used for transmitting either analog or digital information.The bandwidth depends on the thickness of the wire and the distance traveled, but several megabits/sec can be achieved for a few kilometers in many cases. Due to their adequate performance and low cost, twisted pairs are widely used and are likely to remain so for years to come.

Twisted-pair cabling comes in several varieties. The garden variety deployed in many office buildings is called Category 5 cabling, or "Cat 5." A category 5 twisted pair consists of two insulated wires gently twisted together. Four such pairs are typically grouped in a plastic sheath to protect the wires and keep themtogether. This arrangement is shown in Fig. 4.1.

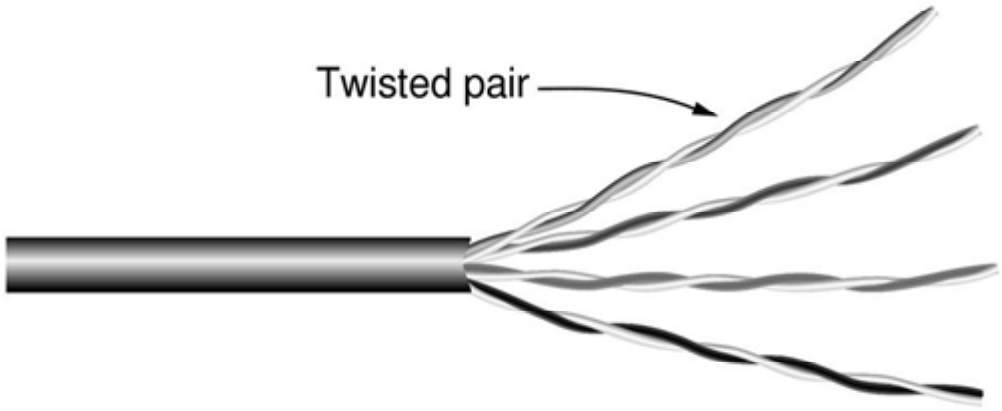


Figure 4.1 Category 5 UTP cable with four twisted pairs.

Different LAN standards may use the twisted pairs differently. For example,100-Mbps Ethernet uses two (out of the four) pairs, one pair for each direction. To reach higher speeds, 1-Gbps Ethernet uses all four pairs in both directions simultaneously;this requires the receiver to factor out the signal that is transmitted locally.

Some general terminology is now in order. Links that can be used in both directions at the same time, like a two-lane road, are called **full-duplex** links. Incontrast, links that can be used in either direction, but only one way at a time, likea single-track railroad line. are called **half-duplex** links. A third category consistsof links that allow traffic in only one direction, like a one-way street. Theyare called **simplex** links.

Returning to twisted pair, Cat 5 replaced earlier Category 3 cables with a similar cable that uses the same connector, but has more twists per meter. More twists result in less crosstalk and a better-quality signal over longer distances,making the cables more suitable for high-speed computer communication, especially 100-Mbps and 1-Gbps Ethernet LANs.

New wiring is more likely to be Category 6 or even Category 7. These categories has more stringent specifications to handle signals with greater bandwidths.Some cables in Category

6 and above are rated for signals of 500 MHz and can support the 10-Gbps links that will soon be deployed.

Through Category 6, these wiring types are referred to as UTP (Unshielded Twisted Pair) as they consist simply of wires and insulators. In contrast to these,Category 7 cables have shielding on the individual twisted pairs, as well as around the entire cable (but inside the plastic protective sheath). Shielding reduces the susceptibility to external interference and crosstalk with other nearby cables to meet demanding performance specifications. The cables are reminiscent of the high-quality, but bulky and expensive shielded twisted pair cables that IBM introduced in the early 1980s, but which did not prove popular outside of IBM installations.Evidently, it is time to try again.

4.3.3 Coaxial Cable

Another common transmission medium is the coaxial cable. It has better shielding and greater bandwidth than unshielded twisted pairs, so it can span longer distances at higher speeds. Two kinds of coaxial cable are widely used. One kind, 50-ohm cable, is commonly used when it is intended for digital transmission from the start. The other kind, 75-ohm cable, is commonly used for analog transmission and cable television. This distinction is based on historical, rather than technical,factors. Starting in the mid-1990s, cable TV operators began to provide Internet access over cable, which has made 75-ohm cable more important for data communication.

A coaxial cable consists of a stiff copper wire as the core, surrounded by an insulating material. The insulator is encased by a cylindrical conductor, often as a closely woven braided mesh. The outer conductor is covered in a protective plastic sheath. A cutaway view of a coaxial cable is shown in Fig. 4.2.

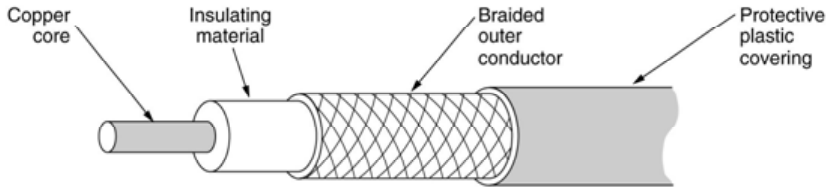


Figure 4.2 A coaxial cable

The construction and shielding of the coaxial cable give it a good combination of high bandwidth and excellent noise immunity. The bandwidth possible depends on the cable quality and length. Modern cables have a bandwidth of up to afew GHz. Coaxial cables used to be widely used within the telephone system for long-distance lines but have now largely been replaced by fiber optics on long-haul routes. Coax is still widely used for cable television and metropolitan area networks.

4.3.4 Power Lines

The telephone and cable television networks are not the only sources of wiring that can be reused for data communication. There is a yet more common kind of wiring: electrical power lines. Power lines deliver electrical power to houses,and electrical wiring within houses distributes the power to electrical outlets.

The use of power lines for data communication is an old idea. Power lines have been used by electricity companies for low-rate communication such as remote metering for many years, as well in the home to control devices (e.g., the X10 standard). In recent years there has been renewed interest in high-rate communication over these lines, both inside the home as a LAN and outside the home for broadband Internet access. We will concentrate on the most common scenario:using electrical wires inside the home.

The convenience of using power lines for networking should be clear. Simply plug a TV and a receiver into the wall, which you must do anyway because they need power, and they can send and receive movies over the electrical wiring. There is no other plug or radio. The data signalis superimposed on the low-frequency power signal (on the active or "hot"wire) as both signals use the wiring at the same time.

The difficulty with using household electrical wiring for a network is that it was designed to distribute power signals. This task is quite different than distributing data signals, at which household wiring does a horrible job. Electrical signalsare sent at 50-60 Hz and the wiring attenuates the much higher frequency(MHz) signals needed for high-rate data communication. The electrical propertiesof the wiring vary from one house to the next and change as appliances are turnedon and off, which causes data signals to bounce around the wiring. Transient currentswhen appliances switch on and off create electrical noise over a wide rangeof frequencies. And without the careful twisting of twisted pairs, electrical wiring acts as a fine antenna, picking up external signals and radiating signals of its own.This behavior means that to meet regulatory requirements, the data signal must exclude licensed frequencies such as the amateur radio bands.

Despite these difficulties, it is practical to send at least 100 Mbps over typical household electrical wiring by using communication schemes that resist impaired frequencies and bursts of errors. Many products use various proprietary standardsfor power-line networking, so international standards are actively under development.

4.3.5 Fiber Optics

Many people in the computer industry take enormous pride in how fast computer technology is improving as it follows Moore's law, which predicts a doubling of the number of transistors per chip roughly every two years (Schaller,1997). The original (1981) IBM PC ran at a clock speed of 4.77 MHz.Twenty eight years later, PCs could run a four-core CPU at 3 GHz. This increase is a gain of a factor of around 2500, or 16 per decade.

In the same period, wide area communication links went from 45 Mbps (a T3 line in the telephone system) to 100 Gbps (a modern long distance line). Thisgain is similarly impressive, more than a factor of 2000 and close to 16 per decade, while at the same time the error rate went from 10?5 per bit to almost zero. Furthermore, single CPUs are beginning to approach physical limits, which is why it is now the number of CPUs that is being increased per chip. In contrast,the achievable bandwidth with fiber technology is in excess of 50,000 Gbps (50Tbps) and we are nowhere near reaching these limits. The current practical limitof around 100 Gbps is due to our inability to convert between electrical and optical signals any faster. To build higher-capacity links, many channels are simply carried in parallel over a single fiber.

In this section we will study fiber optics to learn how that transmission technology works. In the ongoing race between computing and communication, communication may yet win because of fiber optic networks. The implication of this would be essentially infinite bandwidth and a new conventional wisdom that computersare hopelessly slow so that networks should try to avoid computation at all costs, no matter how much bandwidth that wastes. This change will take a while to sink in to a generation of computer scientists and engineers taught to think in terms of the low Shannon limits imposed by copper.

Of course, this scenario does not tell the whole story because it does not include cost. The cost to install fiber over the last mile to reach consumers and bypass the low bandwidth of wires and limited availability of spectrum is tremendous.It also costs more energy to move bits than to compute. We may always have islands of inequities where either computation or communication is essentially free. For example, at the edge of the Internet we throw computation and storage at the problem of compressing and caching content, all to make better useof Internet access links. Within the Internet, we may do the reverse, with companies such as Google moving huge amounts of data across the network to whereit is cheaper to store or compute on it.

Fiber optics are used for long-haul transmission in network backbones, high speed LANs,and high-speed Internet access such as FttH(Fiber to the Home). An optical transmission system has three key components: the light source, the transmission medium, and the detector. Conventionally, a pulse of light indicates a 1 bit andthe absence of light indicates a 0 bit. The transmission medium is an ultra-thin fiber of glass. The detector generates an electrical pulse when light falls on it. By attaching a light source to one end of an optical fiber and a detector to the other,we have a unidirectional data transmission system that accepts an electrical signal,converts and transmits it by light pulses, and then reconverts the output to anelectrical signal at the receiving end.

4.3.5.1 Fiber Cables

Fiber optic cables are similar to coax, except without the braid. Figure 4.3(a) shows a single fiber viewed from the side. At the center is the glass core through which the light propagates. In multimode fibers, the core is typically 50 microns in diameter, about the thickness of a human hair. In single-mode fibers, the core is 8 to 10 microns.

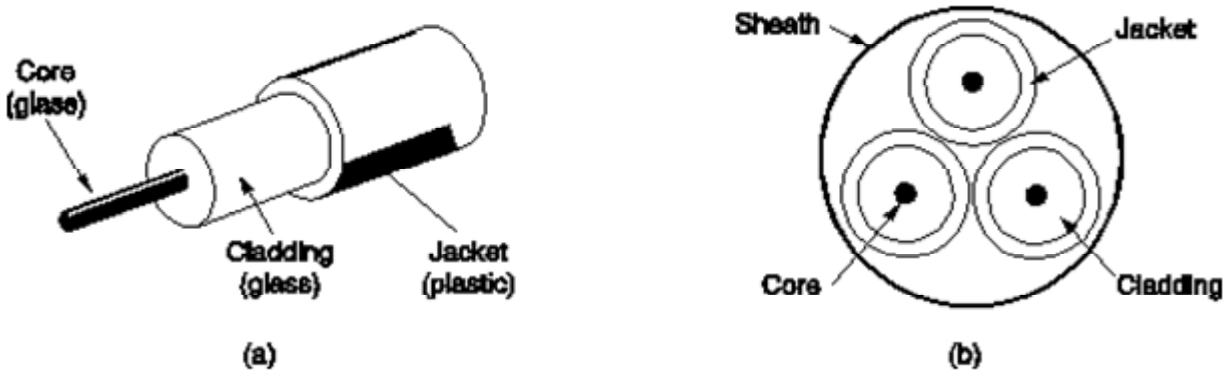


Figure 4.3 (a) Side view of a single fiber. (b) End view of a sheath with three fibers

The core is surrounded by a glass cladding with a lower index of refraction than the core, to keep all the light in the core. Next comes a thin plastic jacket to protect the cladding. Fibers are typically grouped in bundles, protected by sheath. Figure 4.3(b) shows a sheath with three fibers.

Terrestrial fiber sheaths are normally laid in the ground within a meter of thesurface, where they are occasionally subject to attacks by backhoes or gophers.Near the shore, transoceanic fiber sheaths are buried in trenches by a kind of seaplow. In deep water, they just lie on the bottom, where they can be snagged by fishing trawlers or attacked by giant squid.

Fibers can be connected in three different ways. First, they can terminate inconnectors and be plugged into fiber sockets. Connectors lose about 10 to 20% ofthe light, but they make it easy to reconfigure systems.

Second, they can be spliced mechanically. Mechanical splices just lay thetwo carefully cut ends next to each other in a special sleeve and clamp them in place. Alignment can be improved by passing light through the junction and then making small adjustments to maximize the signal. Mechanical splices take trained personnel about 5 minutes and result in a 10% light loss.

Third, two pieces of fiber can be fused (melted) to form a solid connection. A fusion splice is almost as good as a single drawn fiber, but even here, a small amount of attenuation occurs.

For all three kinds of splices, reflections can occur at the point of the splice,and the reflected energy can interfere with the signal.

Two kinds of light sources are typically used to do the signaling. These are LEDs (Light Emitting Diodes) and semiconductor lasers. They have different properties, as shown in Fig. 4.4. They can be tuned in wavelength by inserting Fabry-Perot or Mach-Zehnder interferometers between the source and the fiber.Fabry-Perot interferometers are simple resonant cavities consisting of two parallel mirrors. The light is incident perpendicular to the mirrors. The length of the cavity selects out those wavelengths that fit inside an integral number of times.Mach-Zehnder interferometers separate the light into two beams. The two beams travel slightly different distances. They are recombined at the end and are in phase for only certain wavelengths.

Item	LED	Semiconductor laser
Data rate	Low	High
Mode	Multimode	Multimode or single mode
Distance	Short	Long
Lifetime	Long life	Short life
Temperature sensitivity	Minor	Substantial
Cost	Low cost	Expensive

Figure 4.4 A comparison of semiconductor diodes and LEDs as light sources

The receiving end of an optical fiber consists of a photodiode, which gives off an electrical pulse when struck by light. The response time of photodiodes, which convert the signal from the optical to the electrical domain, limits data rates to about 100 Gbps. Thermal noise is also an issue, so a pulse of light must carry enough energy to be detected. By making the pulses powerful enough, the error rate can be made arbitrarily small.

4.3.5.2 Comparison of Fiber Optics and Copper Wire

It is instructive to compare fiber to copper. Fiber has many advantages. To start with, it can handle much higher bandwidths than copper. This alone would require its use in high-end networks. Due to the low attenuation, repeaters are needed only about every 50 km on long

lines, versus about every 5 km for copper, resulting in a big cost saving. Fiber also has the advantage of not being affected by power surges, electromagnetic interference, or power failures. Nor is it affected by corrosive chemicals in the air, important for harsh factory environments.

Oddly enough, telephone companies like fiber for a different reason: it is thin and lightweight. Many existing cable ducts are completely full, so there is no room to add new capacity. Removing all the copper and replacing it with fiber empties the ducts, and the copper has excellent resale value to copper refiners who see it as very high-grade ore. Also, fiber is much lighter than copper. One thousand twisted pairs 1 km long weigh 8000 kg. Two fibers have more capacityand weigh only 100 kg, which reduces the need for expensive mechanical support systems that must be maintained. For new routes, fiber wins hands down due toits much lower installation cost. Finally, fibers do not leak light and are difficultto tap. These properties give fiber good security against potential wiretappers.

On the downside, fiber is a less familiar technology requiring skills not all engineershave, and fibers can be damaged easily by being bent too much. Since optical transmission is inherently unidirectional, two-way communication requires either two fibers or two frequency bands on one fiber. Finally, fiber interfaces cost more than electrical interfaces. Nevertheless, the future of all fixed data communication over more than short distances is clearly with fiber.

4.4 Summary

The physical layer is the basis of all networks. Nature imposes two fundamental limits on all channels, and these determine their bandwidth. These limits are the Nyquist limit, which deals with noiseless channels, and the Shannon limit,which deals with noisy channels. Transmission media can be guided or unguided. The principal guided media are twisted pair, coaxial cable, and fiber optics.

4.5 Model Questions

1. Write short notes on Coaxial cable.
2. Explain the Guided transmission media.
3. Compare: Fiber Optics and Copper Wire.

LESSON-5
WIRELESS TRANSMISSION

Structure

5.1 Introduction

Our age has given rise to information junkies: people who need to be online all the time. For these mobile users, twisted pair, coax, and fiber optics are of nouse. They need to get their "hits" of data for their laptop, notebook, shirt pocket,palmtop, or wristwatch computers without being tethered to the terrestrial communication infrastructure. For these users, wireless communication is the answer.

In the following sections, we will look at wireless communication in general.It has many other important applications besides providing connectivity to users who want to surf the Web from the beach. Wireless has advantages for even fixed devices in some circumstances. For

example, if running a fiber to a building is difficult due to the terrain (mountains, jungles, swamps, etc.), wireless may be better. It is noteworthy that modern wireless digital communication began in the Hawaiian Islands, where large chunks of Pacific Ocean separated the users fromtheir computer center and the telephone system was inadequate.

5.2 Objectives

- To explain about the various wireless transmission methods.
- To discuss about the digital modulation.
- To discuss about the multiplexing and its types.

5.3 The Electromagnetic Spectrum

When electrons move, they create electromagnetic waves that can propagatethrough space (even in a vacuum). These waves were predicted by the British physicist James Clerk Maxwell in 1865 and first observed by the German physicist Heinrich Hertz in 1887. The number of oscillations per second of awave is called its frequency, f, and is measured in Hz (in honor of HeinrichHertz). The distance between two consecutive maxima (or minima) is called the wavelength, which is universally designated by the Greek letter λ (lambda).

When an antenna of the appropriate size is attached to an electrical circuit, the electromagnetic waves can be broadcast efficiently and received by a receiver some distance away. All wireless communication is based on this principle.

In a vacuum, all electromagnetic waves travel at the same speed, no matter what their frequency. This speed, usually called the speed of light, c, is approximately 3 × 108 m/sec, or about 1 foot (30 cm) per nanosecond. In copper or fiber the speed slows to about 2/3 of this value and becomes slightly frequency dependent.The speed of light is the ultimate speed limit. No object or signal can evermove faster than it.

The fundamental relation between f, λ, and c (in a vacuum) is

$$\lambda f = c$$

Since c is a constant, if we know f, we can find λ, and vice versa. As a rule of thumb, when λ is in meters and f is in MHz, $\lambda f \sim \sim 300$. For example, 100-MHz waves are about 3 meters long, 1000-MHz waves are 0.3 meters long, and 0.1-meter waves have a frequency of 3000 MHz.

The electromagnetic spectrum is shown in Fig. 5.1. The radio, microwave,infrared, and visible light portions of the spectrum can all be used for transmitting information by modulating the amplitude, frequency, or phase of the waves.Ultraviolet light, X-rays, and gamma rays would be even better, due to their higher frequencies, but they are hard to produce and modulate, do not propagate well through buildings, and are dangerous to living things. The bands listed at the bottom of Fig. 5.1 are the official ITU (International Telecommunication Union)names and are based on the wavelengths, so the LF band goes from 1 km to 10 km(approximately 30 kHz to 300 kHz). The terms LF, MF, and HF refer to Low,Medium, and High Frequency, respectively. Clearly, when the names were assigned nobody expected to go above 10 MHz, so the higher bands were later named the Very, Ultra, Super, Extremely, and Tremendously High Frequency bands. Beyond that there are no names, but Incredibly, Astonishingly, and Prodigiously High Frequency (IHF, AHF, and PHF) would sound nice.

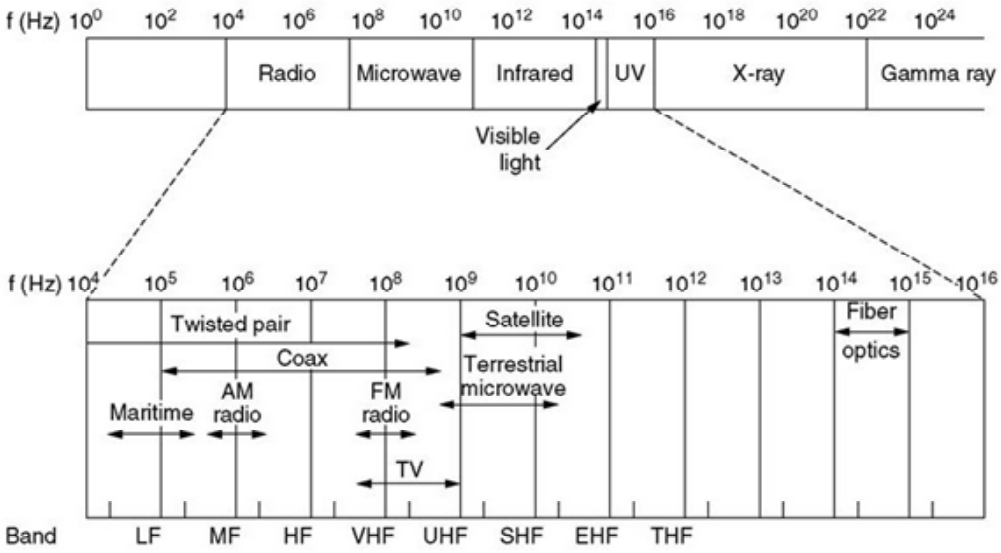


Figure 5.1 The electromagnetic spectrum and its uses for communication

We know from Shannon equation in the previous lesson that the amount of information that a signal such as an electromagnetic wave can carry depends on the received power andis proportional to its bandwidth. From Fig. 5.1 it should now be obvious why networking people like fiber optics so much. Many GHz of bandwidth are available to tap for data transmission in the microwave band, and even more in fiber because it is further to the right in our logarithmic scale.

Most transmissions use a relatively narrow frequency band (i.e., $\Delta f / f << 1$).They concentrate their signals in this narrow band to use the spectrum efficiently and obtain reasonable data rates by transmitting with enough power. However, in some cases, a wider band is used, with three variations. In frequency hopping spread spectrum, the transmitter hops from frequency to frequency hundreds oftimes per second. It is popular for military communication because it makes transmissions hard to detect and next to impossible to jam. It also offers goodresistance to multipath fading and narrowband interference because the receiver will not be stuck on an impaired frequency for long enough to shut down communication.This robustness makes it useful for crowded parts of the spectrum, suchas the ISM bands. This technique is used commercially,for example, in Bluetooth and older versions of 802.11.

A second form of spread spectrum, direct sequence spread spectrum, uses acode sequence to spread the data signal over a wider frequency band. It is widely used commercially as a spectrally efficient way to let multiple signals share the same frequency band. These signals can be given different codes, a method called CDMA (Code Division Multiple Access). This method is shown in contrast with frequency hopping in Fig. 5.2. It forms the basis of 3G mobile phone networks and is also used in GPS (Global Positioning System). Even without different codes, direct sequence spread spectrum,like frequency hopping spread spectrum, can tolerate narrowb and interference and multipath fading because only a fraction of the desired signal is lost.It is used in this role in older 802.11b wireless LANs.

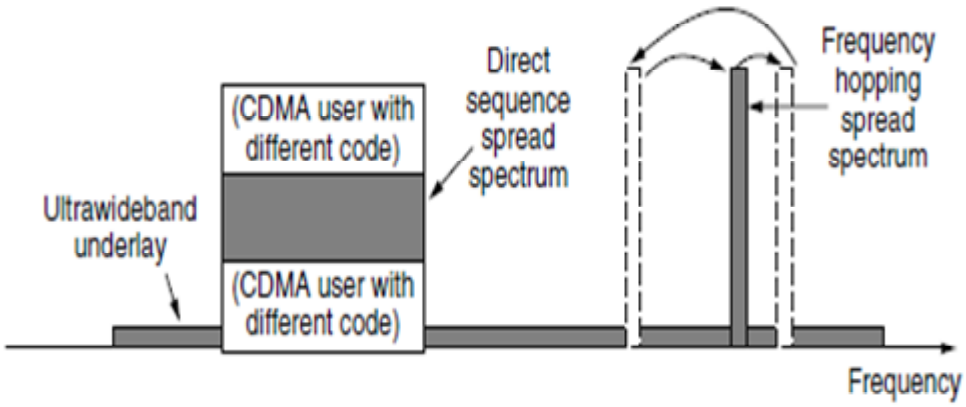


Figure 5.2 Spread spectrum and ultra-wideband (UWB) communication

A third method of communication with a wider band is UWB (Ultra-WideBand) communication. UWB sends a series of rapid pulses, varying their positions to communicate information. The rapid transitions lead to a signal that is spread thinly over a very wide frequency band. UWB is defined as signals that have a bandwidth of at least 500 MHz or at least 20% of the center frequency of their frequency band. UWB is also shown in Fig. 5.2. With this much bandwidth,UWB has the potential to communicate at high rates. Because it is spread across a wide band of frequencies, it can tolerate a substantial amount of relatively strong interference from other narrowband signals. Just as importantly, since UWB has very little energy at any given frequency when used for short-range transmission, it does not cause harmful interference to those other narrowband radio signals. It is said to underlay the other signals. This peaceful coexistence has led to its application in wireless PANs that run at up to 1 Gbps, although commercial success has been mixed. It can also be used for imaging through solid objects(ground, walls, and bodies) or as part of precise location systems.We will now discuss how the various parts of the electromagnetic spectrum of Fig. 5.2 are used, starting with radio. We will assume that all transmissions usea narrow frequency band.

5.4 Radio Transmission

Radio frequency (RF) waves are easy to generate, can travel long distances,and can penetrate buildings easily, so they are widely used for communication,both indoors and outdoors. Radio waves also are omnidirectional, meaning thatthey travel in all directions from the source, so the transmitter and receiver do not have to be carefully aligned physically.

The properties of radio waves are frequency dependent. At low frequencies,radio waves pass through obstacles well, but the power falls off sharply with distance from the source-at least as fast as 1/r 2 in air-as the signal energy is spread more thinly over a larger surface. This attenuation is called path loss. At high frequencies, radio waves tend to travel in straight lines and bounce off obstacles.Path loss still reduces power, though the received signal can depend strongly on reflections as well. High-frequency radio waves are also absorbed by rain andother obstacles to a larger extent than are low-frequency ones. At all frequencies,radio waves are subject to interference from motors and other electrical equipment.

It is interesting to compare the attenuation of radio waves to that of signals inguided media. With fiber, coax and twisted pair, the signal drops by the same fraction per unit distance, for example 20 dB per 100m for twisted pair. With radio, the signal drops by the same fraction as the distance doubles, for example 6dB per doubling in free space. This behavior means that radio waves can travellong distances, and interference between users is a problem. For this reason, all governments tightly regulate the use of radio transmitters, with few notable exceptions.

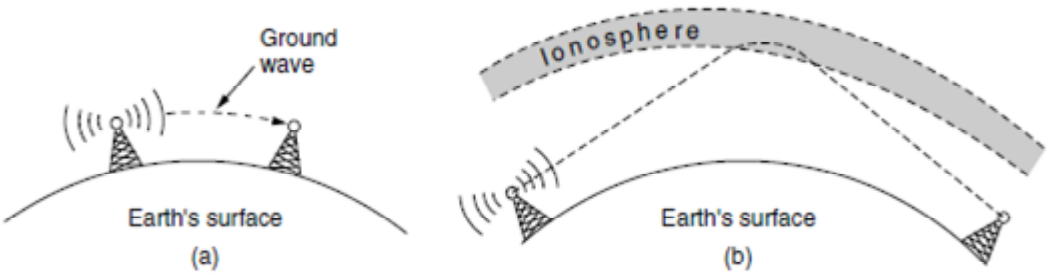


Figure 5.3(a) In the VLF, LF, and MF bands, radio waves follow the curvatureof the earth. (b) In the HF band, they bounce off the ionosphere.

In the VLF, LF, and MF bands, radio waves follow the ground, as illustrated in Fig. 5.3 (a). These waves can be detected for perhaps 1000 km at the lower frequencies, less at the higher ones. AM radio broadcasting uses the MF band,which is why the ground waves from Boston AM radio stations cannot be heardeasily in New York. Radio waves in these bands pass through buildings easily,which is why portable radios work indoors. The main problem with using these bands for data communication is their low bandwidth.

In the HF and VHF bands, the ground waves tend to be absorbed by the earth.However, the waves that reach the ionosphere, a layer of charged particles circling the earth at a height of 100 to 500 km, are refracted by it and sent back toearth, as shown in Fig. 5.3(b). Under certain atmospheric conditions, the signalscan bounce several times. Amateur radio operators (hams) use these bands to talklong distance. The military also communicate in the HF and VHF bands.

5.5 Microwave Transmission

Above 100 MHz, the waves travel in nearly straight lines and can therefore be narrowly focused. Concentrating all the energy into a small beam by means of a parabolic antenna (like the familiar satellite TV dish) gives a much higher signal-to-noise ratio, but the transmitting and receiving antennas must be accurately aligned with each other. In addition, this directionality allows multiple transmitters lined up in a row to communicate with multiple receivers in a row without interference, provided some minimum spacing rules are observed. Before fiber optics, for decades these microwaves formed the heart of the long-distance telephone transmission system. In fact, MCI, one of AT&T's first competitors after it was deregulated, built its entire system with microwave communications passing between towers tens of kilometers apart. Even the company's name reflected this(MCI stood for Microwave Communications, Inc.). MCI has since gone over to fiber and through a long series of corporate mergers and bankruptcies in the telecommunications shuffle has become part of Verizon.

Microwaves travel in a straight line, so if the towers are too far apart, the earth will get in the way. Thus, repeaters are needed periodically. The higher the towers are, the farther apart they can be. The distance between repeaters goes up very roughly with the square root of the tower height. For 100-meter-high towers, repeaters can be 80 km apart.

Unlike radio waves at lower frequencies, microwaves do not pass through buildings well. In addition, even though the beam may be well focused at the transmitter, there is still some divergence in space. Some waves may be refracted off low-lying atmospheric layers and may take slightly longer to arrive than the direct waves. The delayed waves may arrive out of phase with the direct wave and thus cancel the signal. This effect is called multipath fading and is often aserious problem. It is weather and frequency dependent. Some operators keep10% of

their channels idle as spares to switch on when multipath fading temporarilywipes out some frequency band.

The demand for more and more spectrum drives operators to yet higher frequencies.Bands up to 10 GHz are now in routine use, but at about 4 GHz a new problem sets in: absorption by water. These waves are only a few centimeters long and are absorbed by rain. This effect would be fine if one were planning to build a huge outdoor microwave oven for roasting passing birds, but for communication it is a severe problem. As with multipath fading, the only solution is toshut off links that are being rained on and route around them.

In summary, microwave communication is so widely used for long-distance telephone communication, mobile phones, television distribution, and other purposes that a severe shortage of spectrum has developed. It has several key advantages over fiber. The main one is that no right of way is needed to lay down cables. By buying a small plot of ground every 50 km and putting a microwave tower on it, one can bypass the telephone system entirely. This is how MCI managed to get started as a new long-distance telephone company so quickly.

Microwave is also relatively inexpensive. Putting up two simple towers and putting antennas on each one may be cheaper than burying 50 km of fiber through a congested urban areaor up over a mountain, and it may also be cheaper than leasing the telephone company's fiber, especially if the telephone company has not yet even fully paid forthe copper it ripped out when it put in the fiber.

5.6 Infrared Transmission

Unguided infrared waves are widely used for short-range communication.The remote controls used for televisions, VCRs, and stereos all use infrared communication.They are relatively directional, cheap, and easy to build but have amajor drawback: they do not pass through solid objects. In general, as we go from long-wave radio toward visible light, the waves behave more and more like light and less and less like radio.

On the other hand, the fact that infrared waves do not pass through solid walls well is also a plus. It means that an infrared system in one room of a building will not interfere with a similar system in adjacent rooms or buildings: you cannot control your neighbor's television with your

remote control. Furthermore, security of infrared systems against eavesdropping is better than that of radio systems precisely for this reason. Therefore, no government license is needed to operate an infrared system, in contrast to radio systems, which must be licensed outside the ISM bands. Infrared communication has a limited use on the desktop, for example,to connect notebook computers and printers with the IrDA (Infrared Data Association) standard, but it is not a major player in the communication game.

5.7 Light Transmission

Unguided optical signaling or free-space optics has been in use for centuries.A more modern application is to connect the LANs in two buildings via lasers mounted on their rooftops. Optical signaling using lasers is inherently unidirectional, so each end needs its own laser and its own photo detector.This scheme offers very high bandwidth at very low cost and is relatively secure because it is difficult to tap a narrow laser beam. It is also relatively easy to install and, unlike microwave transmission, does not require an FCC license.

The laser's strength, a very narrow beam, is also its weakness here. Aiming a laser beam 1 mm wide at a target the size of a pin head 500 meters away requires the marksmanship of a latter-day Annie Oakley. Usually, lenses are put into the system to defocus the beam slightly. To add to the difficulty, wind and temperature changes can distort the beam and laser beams also cannot penetrate rain or thick fog, although they normally work well on sunny days. However, many of these factors are not an issue when the use is to connect two spacecraft.

Unguided optical communication may seem like an exotic networking technology today, but it might soon become much more prevalent. We are surrounded by cameras (that sense light) and displays (that emit light using LEDs and other technology). Data communication can be layered on top of these displays by encoding information in the pattern at which LEDs turn on and off that is below the threshold of human perception. Communicating with visible light in this way is inherently safe and creates a low-speed network in the immediate vicinity of the display. This could enable all sorts of fanciful ubiquitous computing scenarios.The flashing lights on emergency vehicles might alert nearby traffic lights and vehicles to help clear a path. Informational signs might broadcast maps. Even festive lights might broadcast songs that are synchronized with their display.

5.8 Communication Satellites

Further progress in the celestial communication field had to wait until the first communication satellite was launched. The key difference between an artificial satellite and a real one is that the artificial one can amplify the signals before sending them back, turning a strange curiosity into a powerful communication system.

Communication satellites have some interesting properties that make them attractive for many applications. In its simplest form, a communication satellite can be thought of as a big microwave repeater in the sky. It contains several transponders, each of which listens to some portion of the spectrum, amplifies the incoming signal, and then rebroadcasts it at another frequency to avoid interference with the incoming signal. This mode of operation is known as a **bentpipe**. Digital processing can be added to separately manipulate or redirect datastreams in the overall band, or digital information can even be received by the satellite and rebroadcast. Regenerating signals in this way improves performance compared to a bent pipe because the satellite does not amplify noise in the upward signal. The downward beams can be broad, covering a substantial fraction of the earth's surface, or narrow, covering an area only hundreds of kilometers in diameter.

According to Kepler's law, the orbital period of a satellite varies as the radius of the orbit to the 3/2 power. The higher the satellite, the longer the period. Near the surface of the earth, the period is about 90 minutes. Consequently, low-orbit satellites pass out of view fairly quickly, so many of them are needed to provide continuous coverage and ground antennas must track them. At an altitude of about 35,800 km, the period is 24 hours. At an altitude of 384,000 km, the periodis about one month, as anyone who has observed the moon regularly can testify.

A satellite's period is important, but it is not the only issue in determining where to place it. Another issue is the presence of the Van Allen belts, layers of highly charged particles trapped by the earth's magnetic field. Any satellite flying within them would be destroyed fairly quickly by the particles. These factors leadto three regions in which satellites can be placed safely. Below we will briefly describe the satellites that inhabit each of these regions.

5.8.1 Geostationary Satellites

In 1945, the science fiction writer Arthur C. Clarke calculated that a satelliteat an altitude of 35,800 km in a circular equatorial orbit would appear to remain motionless in the sky, so it would not need to be tracked (Clarke, 1945). He wenton to describe a complete communication system that used these (manned) geostationary satellites, including the orbits, solar panels, radio frequencies, and launch procedures. Unfortunately, he concluded that satellites were impractical due to the impossibility of putting power-hungry, fragile vacuum tube amplifiers into orbit, so he never pursued this idea further, although he wrote some science fiction stories about it.

The invention of the transistor changed all that, and the first artificial communication satellite, Telstar, was launched in July 1962. Since then, communication satellites have become a multibillion dollar business and the only aspect of outerspace that has become highly profitable. These high-flying satellites are often called GEO (Geostationary Earth Orbit) satellites.

With current technology, it is unwise to have geostationary satellites spaced much closer than 2 degrees in the 360-degree equatorial plane, to avoid interference.With a spacing of 2 degrees, there can only be 360/2 = 180 of these satellites in the sky at once. However, each transponder can use multiple frequencies and polarizations to increase the available bandwidth.To prevent total chaos in the sky, orbit slot allocation is done by ITU. This process is highly political, with countries barely out of the stone age demanding "their" orbit slots (for the purpose of leasing them to the highest bidder). Other countries, however, maintain that national property rights do not extend up to the moon and that no country has a legal right to the orbit slots above its territory. To add to the fight, commercial telecommunication is not the only application. Television broadcasters, governments, and the military also want a piece of the orbiting pie.

Modern satellites can be quite large, weighing over 5000 kg and consuming several kilowatts of electric power produced by the solar panels. The effects of solar, lunar, and planetary gravity tend to move them away from their assigned orbit slots and orientations, an effect countered by on-board rocket motors. This fine-tuning activity is called station keeping. However, when the fuel for themotors has been exhausted (typically after about 10 years) the satellite drifts and tumbles helplessly, so it has to be turned off. Eventually, the orbit decays and the satellite reenters the atmosphere and burns up (or very rarely crashes to earth).

Orbit slots are not the only bone of contention. Frequencies are an issue, too,because the downlink transmissions interfere with existing microwave users.Consequently, ITU has allocated certain frequency bands to satellite users. The main ones are listed in Fig. 5.4. The C band was the first to be designated for commercial satellite traffic. Two frequency ranges are assigned in it, the lower one for downlink traffic (from the satellite) and the upper one for uplink traffic (tothe satellite). To allow traffic to go both ways at the same time, two channels are required. These channels are already overcrowded because they are also used by the common carriers for terrestrial microwave links. The L and S bands were added by international agreement in 2000. However, they are narrow and also crowded.

Band	Downlink	Uplink	Bandwidth	Problems
L	1.5 GHz	1.6 GHz	15 MHz	Low bandwidth; crowded
S	1.9 GHz	2.2 GHz	70 MHz	Low bandwidth; crowded
C	4.0 GHz	6.0 GHz	500 MHz	Terrestrial interference
Ku	11 GHz	14 GHz	500 MHz	Rain
Ka	20 GHz	30 GHz	3500 MHz	Rain, equipment cost

Figure 5.4 The principal satellite bands

The next-highest band available to commercial telecommunication carriers isthe Ku (K under) band. This band is not (yet) congested, and at its higher frequencies,satellites can be spaced as close as 1 degree. However, another problem exists: rain. Water absorbs these short microwaves well. Fortunately, heavy storms are usually localized, so using several widely separated ground stations insteadof just one circumvents the problem, but at the price of extra antennas, extra cables, and extra electronics to enable rapid switching between stations. Bandwidth has also been allocated in the Ka (K above) band for commercial satellite traffic, but the equipment needed to use it is expensive. In addition to these commercial bands, many government and military bands also exist.

A modern satellite has around 40 transponders, most often with a 36-MHz bandwidth. Usually, each transponder operates as a bent pipe, but recent satellites have some on-board processing capacity, allowing more sophisticated operation.In the earliest satellites, the division

of the transponders into channels was static:the bandwidth was simply split up into fixed frequency bands. Nowadays, each transponder beam is divided into time slots, with various users taking turns.

The first geostationary satellites had a single spatial beam that illuminatedabout 1/3 of the earth's surface, called its footprint. With the enormous decline in the price, size, and power requirements of microelectronics, a much more sophisticated broadcasting strategy has become possible. Each satellite is equipped with multiple antennas and multiple transponders. Each downward beam can be focused on a small geographical area, so multiple upward and downward transmissions can take place simultaneously. Typically, these so-called spot beams are elliptically shaped, and can be as small as a few hundred km in diameter.A communication satellite for the United States typically has one wide beam for the contiguous 48 states, plus spot beams for Alaska and Hawaii.

A recent development in the communication satellite world is the development of low-cost microstations, sometimes called VSATs (Very Small ApertureTerminals) (Abramson, 2000). These tiny terminals have 1-meter or smaller antennas(versus 10 m for a standard GEO antenna) and can put out about 1 watt of power. The uplink is generally good for up to 1 Mbps, but the downlink is often up to several megabits/sec. Direct broadcast satellite television uses this technology for one-way transmission.

In many VSAT systems, the microstations do not have enough power to communicate directly with one another (via the satellite, of course). Instead, a special ground station, the hub, with a large, high-gain antenna is needed to relay traffic between VSATs, as shown in Fig. 5.5. In this mode of operation, either the sender or the receiver has a large antenna and a powerful amplifier. The trade-off is a longer delay in return for having cheaper end-user stations.

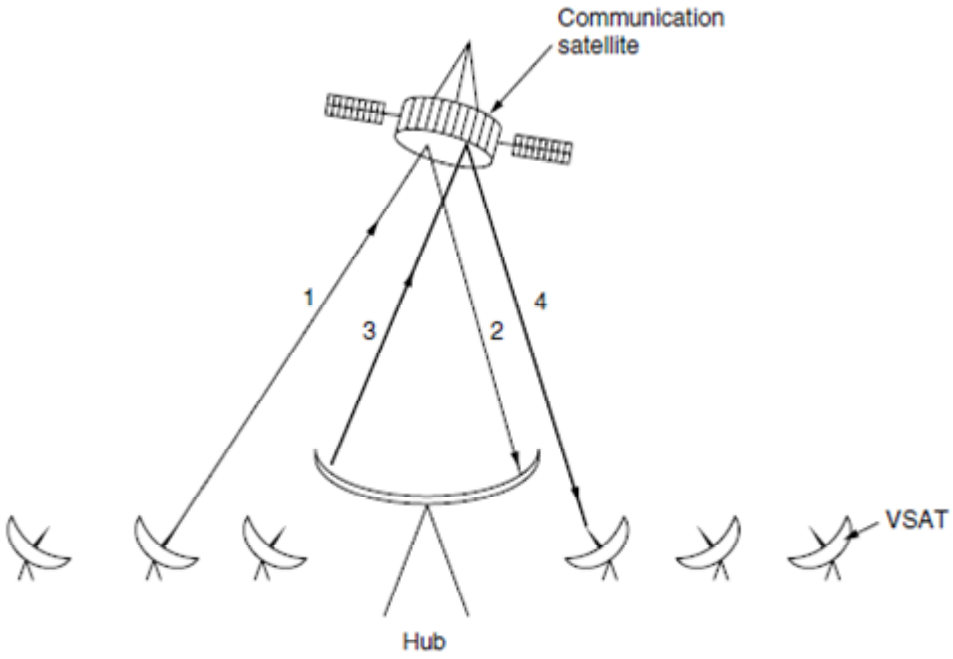


Figure 5.5 VSATs using a hub

VSATs have great potential in rural areas. It is not widely appreciated, but over half the world's population lives more than hour's walk from the nearest telephone. Stringing telephone wires to thousands of small villages is far beyond the budgets of most Third World governments, but installing 1-meter VSATdishes powered by solar cells is often feasible. VSATs provide the technology that will wire the world.

Communication satellites have several properties that are radically different from terrestrial point-to-point links. To begin with, even though signals to and from a satellite travel at the speed of light (nearly 300,000 km/sec), the longround-trip distance introduces a substantial delay for GEO satellites. Depending on the distance between the user and the ground station and the elevation of the satellite above the horizon, the end-to-end transit time is between 250 and 300 msec. A typical value is 270 msec (540 msec for a VSAT system with a hub).

For comparison purposes, terrestrial microwave links have a propagation delay of roughly 3 μsec/km, and coaxial cable or fiber optic links have a delay ofapproximately 5 μsec/km. The latter are slower than the former because electromagnetic signals travel faster in air than in solid materials.

Another important property of satellites is that they are inherently broadcast media. It does not cost more to send a message to thousands of stations within a transponder's footprint than it does to send to one. For some applications, this property is very useful. For example, one could imagine a satellite broadcasting popular Web pages to the caches of a large number of computers spread over awide area. Even when broadcasting can be simulated with point-to-point lines,satellite broadcasting may be much cheaper. On the other hand, from a privacy point of view, satellites are a complete disaster: everybody can hear everything.Encryption is essential when security is required.

Satellites also have the property that the cost of transmitting a message is independent of the distance traversed. A call across the ocean costs no more to service than a call across the street. Satellites also have excellent error rates and can be deployed almost instantly, a major consideration for disaster response and military communication.

5.8.2 Medium-Earth Orbit Satellites

At much lower altitudes, between the two Van Allen belts, we find the MEO(Medium-Earth Orbit) satellites. As viewed from the earth, these drift slowly in longitude, taking something like 6 hours to circle the earth. Accordingly, they must be tracked as they move through the sky. Because they are lower than the GEOs, they have a smaller footprint on the ground and require less powerful transmitters to reach them. Currently they are used for navigation systems rather than telecommunications, so we will not examine them further here. The constellation of roughly 30 GPS (Global Positioning System) satellites orbiting at about 20,200 km are examples of MEO satellites.

5.8.3 Low-Earth Earth Orbit Satellites

Moving down in altitude, we come to the LEO (Low-Earth Orbit) satellites.Due to their rapid motion, large numbers of them are needed for a complete system.On the other hand, because the satellites are so close to the earth, the groundstations do not need much power, and the round-trip delay is only a few milliseconds.The launch cost is substantially cheaper too. In this section we will examine two examples of satellite constellations for voice service, Iridium and Globalstar.

For the first 30 years of the satellite era, low-orbit satellites were rarely used because they zip into and out of view so quickly. In 1990, Motorola broke new ground by filing an application with the FCC asking for permission to launch 77 low-orbit satellites for the Iridium project (element 77 is iridium). The plan waslater revised to use only 66 satellites, so the project should have been renamed Dysprosium (element 66), but that probably sounded too much like a disease. The idea was that as soon as one satellite went out of view, another would replace it.This proposal set off a feeding frenzy among other communication companies.All of a sudden, everyone wanted to launch a chain of low-orbit satellites.

Satellites continue to be launched at a rate of around 20 per year, including ever-larger satellites that now weigh over 5000 kilograms. But there are also very small satellites for the more budget-conscious organization. To make space research more accessible, academics from Cal Poly and Stanford got together in1999 to define a standard for miniature satellites and an associated launcher that would greatly lower launch costs (Nugent et al., 2008). CubeSats are satellites inunits of 10 cm × 10 cm × 10 cm cubes, each weighing no more than 1 kilogram,that can be launched for as little as $40,000 each. The launcher flies as a secondary payload on commercial space missions. It is basically a tube that takes upto three units of cubesats and uses springs to release them into orbit. Roughly 20 cubesats have launched so far, with many more in the works. Most of them communicate with ground stations on the UHF and VHF bands.

5.8.4 Satellites Versus Fiber

A comparison between satellite communication and terrestrial communicationis instructive. As recently as 25 years ago, a case could be made that the future of communication lay with communication satellites. After all, the telephone system had changed little in the previous 100 years and showed no signs of changing inthe next 100 years. This glacial movement was caused in no small part by the regulatory environment in which the telephone companies were expected to provide good voice service at reasonable prices (which they did), and in return got a guaranteed profit on their investment. For people with data to transmit, 1200-bpsmodems were available.

Nevertheless, communication satellites have some major niche markets that fiber does not (and, sometimes, cannot) address. First, when rapid deployment is critical, satellites win

easily. A quick response is useful for military communication systems in times of war and disaster response in times of peace. Following the massive December 2004 Sumatra earthquake and subsequent tsunami, for example,communications satellites were able to restore communications to first responders within 24 hours. This rapid response was possible because there is a developed satellite service provider market in which large players, such as Intelsat with over 50 satellites, can rent out capacity pretty much anywhere it is needed.For customers served by existing satellite networks, a VSAT can be set up easilyand quickly to provide a megabit/sec link to elsewhere in the world.

A second niche is for communication in places where the terrestrial infrastructure is poorly developed. Many people nowadays want to communicate everywhere they go. Mobile phone networks cover those locations with good population density, but do not do an adequate job in other places (e.g., at sea or inthe desert). Conversely, Iridium provides voice service everywhere on Earth,even at the South Pole. Terrestrial infrastructure can also be expensive to install,depending on the terrain and necessary rights of way. Indonesia, for example, hasits own satellite for domestic telephone traffic. Launching one satellite was cheaper than stringing thousands of undersea cables among the 13,677 islands in the archipelago.

A third niche is when broadcasting is essential. A message sent by satellite can be received by thousands of ground stations at once. Satellites are used to distribute much network TV programming to local stations for this reason. There is now a large market for satellite broadcasts of digital TV and radio directly to endusers with satellite receivers in their homes and cars. All sorts of other content can be broadcast too. For example, an organization transmitting a stream of stock, bond, or commodity prices to thousands of dealers might find a satellite system to be much cheaper than simulating broadcasting on the ground.

In short, it looks like the mainstream communication of the future will be terrestrial fiber optics combined with cellular radio, but for some specialized uses, satellites are better. However, there is one caveat that applies to all of this:economics. Although fiber offers more bandwidth, it is conceivable that terrestrial and satellite communication could compete aggressively on price. If advances in technology radically cut the cost of deploying a satellite (e.g., if somefuture space vehicle can toss out dozens of satellites on one launch) or low-orbit satellites catch on in a big way, it is not certain that fiber will win all markets.

5.9 Digital Modulation

Modulation is the process of transforming a carrier signal so that it can carry the information of a message signal. It superimposes the contents of the message signal over a high-frequency carrier signal, which is then transmitted over communication channels.Modulation can be of two types: (1) Analog Modulation and (2) Digital Modulation. Wires and wireless channels carry analog signals such as continuously varying voltage, light intensity, or sound intensity. To send digital information, we must devise analog signals to represent bits. The process of converting between bits and signals that represent them is called digital modulation.

We will start with schemes that directly convert bits into a signal. These schemes result in baseband transmission, in which the signal occupies frequencies from zero up to a maximum that depends on the signaling rate. It is common for wires. Then we will consider schemes that regulate the amplitude, phase, or frequency of a carrier signal to convey bits. These schemes result in passband transmission, in which the signal occupies a band of frequencies around the frequency of the carrier signal. It is common for wireless and optical channels for which the signals must reside in a given frequency band.The following Fig. 5.6 illustrates the different digital modulation schemes:

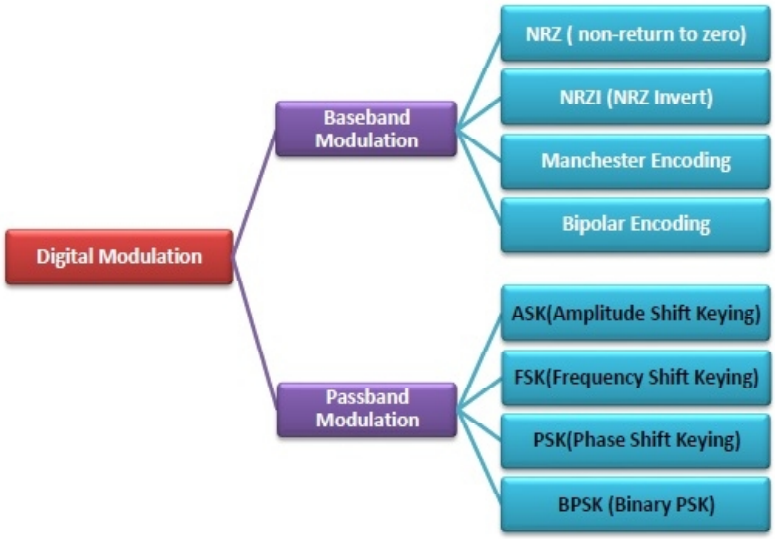


Fig. 5.6 Digital Modulation Schemes

5.10 Multiplexing

Channels are often shared by multiple signals. After all, it is much more convenient to use a single wire to carry several signals than to install a wire for every signal. This kind of sharing is called multiplexing. It can be accomplished in several different ways. We will present methods for time, frequency, and code division multiplexing. The multiplexing techniques we describe in this section are all widely used for wires, fiber, terrestrial wireless, and satellite channels. In the following sections, we will look at examples of networks to see them in action.

5.10.1 Frequency Division Multiplexing

The modulation schemes send one signal to convey bits along a wired or wireless link. However, economies of scale play an important role in how we use networks. It costs essentially the same amount of money to install and maintain a high-bandwidth transmission line as a low-bandwidth line betweentwo different offices. Consequently, multiplexingschemes have been developed to share lines among many signals.

FDM (Frequency Division Multiplexing) takes advantage of passband transmission to share a channel. It divides the spectrum into frequency bands, with each user having exclusive possession of some band in which to send their signal.AM radio broadcasting illustrates FDM. The allocated spectrum is about 1 MHz,roughly 500 to 1500 kHz. Different frequencies are allocated to different logical channels (stations), each operating in a portion of the spectrum, with the interchannel separation great enough to prevent interference.

For a more detailed example, in Fig. 5.7 we show three voice-grade telephone channels multiplexed using FDM. Filters limit the usable bandwidth to about 3100 Hz per voice-grade channel. When many channels are multiplexed together,4000 Hz is allocated per channel. The excess is called a guard band. It keeps the channels well separated. First the voice channels are raised in frequency,each by a different amount. Then they can be combined because no two channels now occupy the same portion of the spectrum. Notice that even though there are gaps between the channels thanks to the guard bands, there is some overlap between adjacent

channels. The overlap is there because real filters do not have ideal sharp edges. This means that a strong spike at the edge of one channel will be felt in the adjacent one as nonthermal noise.

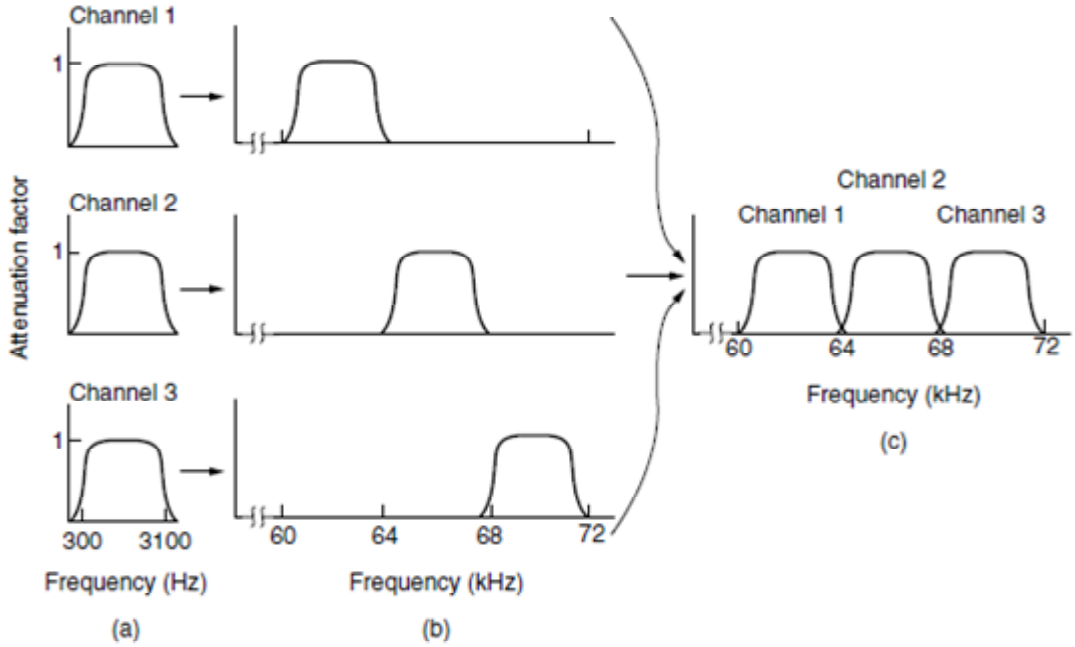


Figure 5.7 Frequency division multiplexing. (a) The original bandwidths. (b) The bandwidths raised in frequency. (c) The multiplexed channel.

This scheme has been used to multiplex calls in the telephone system for many years, but multiplexing in time is now preferred instead. However, FDM continues to be used in telephone networks, as well as cellular, terrestrial wireless,and satellite networks at a higher level of granularity.

5.10.2 Time Division Multiplexing

An alternative to FDM is TDM (Time Division Multiplexing). Here, the users take turns (in a round-robin fashion), each one periodically getting the entire bandwidth for a little burst of time. An example of three streams being multiplexed with TDM is shown in Fig. 5.8. Bits from each input stream are taken ina fixed time slot and output to the aggregate stream. This stream runs at the sumrate of the individual streams. For this to work, the streams must be synchronizedin time. Small intervals of guard time analogous to a frequency guard band may be added to accommodate small timing variations.

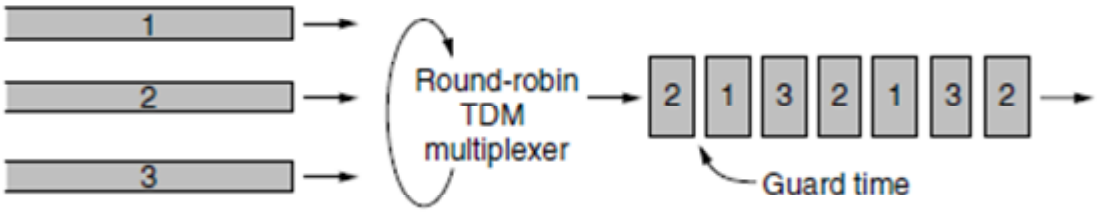


Figure 5.8 Time Division Multiplexing (TDM).

TDM is used widely as part of the telephone and cellular networks. To avoid one point of confusion, let us be clear that it is quite different from the alternative STDM (Statistical Time Division Multiplexing). The prefix "statistical" is added to indicate that the individual streams contribute to the multiplexed stream not on a fixed schedule, but according to the statistics of their demand. STDM is packet switching by another name.

5.10.3 Code Division Multiplexing

There is a third kind of multiplexing that works in a completely different way than FDM and TDM. CDM (Code Division Multiplexing) is a form of spread spectrum communication in which a narrowband signal is spread out over awider frequency band. This can make it more tolerant of interference, as well asallowing multiple signals from different users to share the same frequency band.Because code division multiplexing is mostly used for the latter purpose it is commonly called CDMA (Code Division Multiple Access).

CDMA allows each station to transmit over the entire frequency spectrum all the time. Multiple simultaneous transmissions are separated using coding theory.Before getting into the algorithm, let us consider an analogy: an airport lounge with many pairs of people conversing. TDM is comparable to pairs of people inthe room taking turns speaking. FDM is comparable to the pairs of people speaking at different pitches, some high-pitched and some low-pitched such that each pair can hold its own conversation at the same time as but independently of the others. CDMA is comparable to each pair of people talking at once, but in a different language. The French-speaking couple just hones in on the French, rejecting everything that is not French as noise. Thus, the key to CDMA is to be able to extract the desired signal while rejecting everything else as random noise. A somewhat simplified description of CDMA follows.

In CDMA, each bit time is subdivided into m short intervals called chips.Typically, there are 64 or 128 chips per bit, but in the example given here we will use 8 chips/bit for simplicity. Each station is assigned a unique m-bit code called a chip sequence. For pedagogical purposes, it is convenient to use a bipolar notation to write these codes as sequences of –1 and +1. We will show chip sequences in parentheses.

To transmit a 1 bit, a station sends its chip sequence. To transmit a 0 bit, itsends the negation of its chip sequence. No other patterns are permitted. Thus,for m = 8, if station A is assigned the chip sequence (–1 –1 –1 +1 +1 –1 +1 +1), itcan send a 1 bit by transmiting the chip sequence and a 0 by transmitting(+1 +1 +1 –1 –1 +1 –1 –1). It is really signals with these voltage levels that are sent, but it is sufficient for us to think in terms of the sequences.In Fig. 5.9 (a) and (b) we show the chip sequences assigned to four example stations and the signals that they represent.

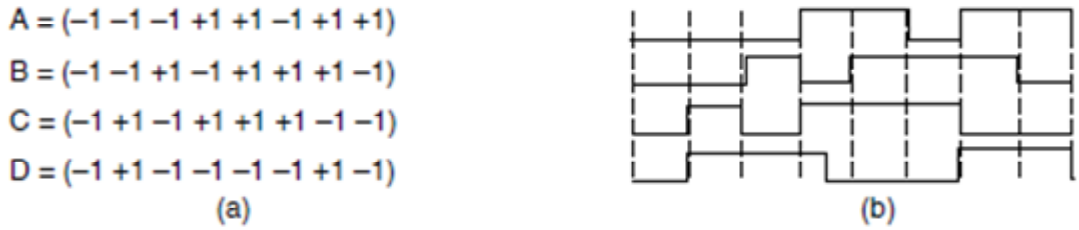


Figure 5.9(a) Chip sequences for four stations. (b) Signals the sequences represent

5.11 Summary

Unguided media include terrestrial radio, microwaves, infrared, lasers through the air, and satellites.Digital modulation methods send bits over guided and unguided media as analogsignals. Line codes operate at baseband, and signals can be placed in a passband by modulating the amplitude, frequency, and phase of a carrier. Channels can be shared between users with time, frequency and code division multiplexing.

5.12 Model Questions

1. Explain the merits of Geostationary satellites.
2. Write a short notes on Radio Transmission.

3. Describe the various Multiplexing techniques.
4. Write a note on Low-Earth Orbit Satellites.
5. Explain about Wireless Transmission.

LESSON-6
THE PUBLIC SWITCHED TELEPHONE NETWORK

Structure

6.1 Introduction

When two computers owned by the same company or organization and located close to each other need to communicate, it is often easiest just to run a cable between them. But, when the distances are large orthere are many computers or the cables have to pass through a public road or other public right of way, the costs of running private cables are usually prohibitive. Consequently, the network designers must rely on the existing telecommunication facilities.

The PSTN (Public Switched Telephone Network),were usually designed many years ago, with a completely different goal in mind: transmitting the human voice in a more-or-less recognizable form. Their suitability for use in computer-computer communication is often marginal at best. The telephone system is tightly intertwined with (wide area)computer networks. Thelimiting factor for networking purposes turns out to be the "last mile" over which customers connect, not the trunks and switches inside the telephone network. Computer systems designers used to working with systems that give at least three orders of magnitude better performance have devoted much time and effort to figure out how to use the telephone network efficiently.In the following sections we will describe the telephone system and show how it works.

6.2 Objectives

- To explain about the structure of the telephone system.
- To discuss about the trunks and multiplexing.
- To discuss about the switching and its types.

6.3 Structure of the Telephone System

Soon after Alexander Graham Bell patented the telephone in 1876, there was an enormous demand for his new invention. The initial market was for the sale of telephones, which came in pairs.It was up to the customer to string a single wire between them. If a telephone owner wanted to talk to n other telephone owners, separate wires had to be strungto all n houses. Within a year, the cities were covered with wires passing overhouses and trees in a wild jumble. It became immediately obvious that the model of connecting every telephone to every other telephone, as shown in Fig. 6.1 (a),was not going to work.

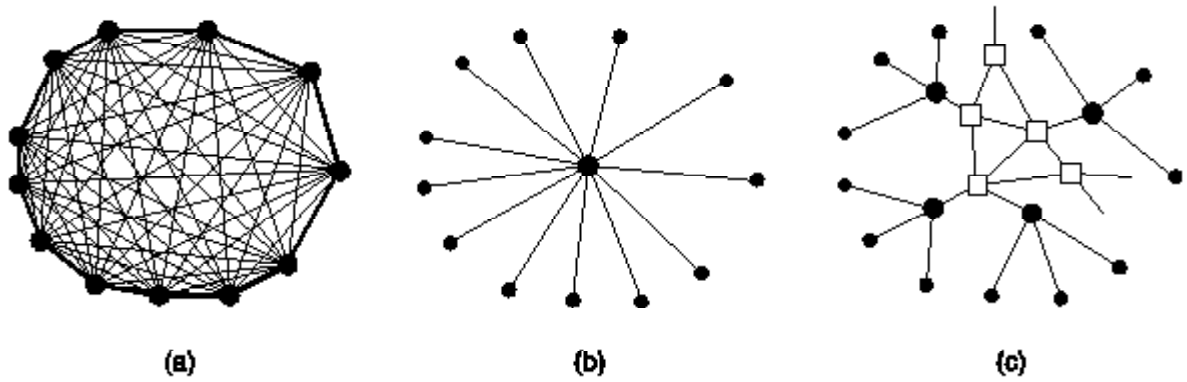


Figure 6.1 (a) Fully interconnected network. (b) Centralized switch. (c) Two-level hierarchy

To his credit, Bell saw this problem early on and formed the Bell Telephone Company, which opened its first switching office (in New Haven, Connecticut) in1878. The company ran a wire to each customer's house or office. To make a call, the customer would crank the phone to make a ringing sound in the telephone company office to attract the attention of an operator, who would then manually connect the caller to the callee by using a short jumper cable to connect the caller to the callee. The model of a single switching office is illustrated in Fig. 6.1 (b).

Pretty soon, Bell System switching offices were springing up everywhere and people wanted to make long-distance calls between cities, so the Bell System began to connect the switching offices. The original problem soon returned: to connect every switching office to every other switching office by means of a wire between them quickly became unmanageable, so second-level switching offices were invented. After a while, multiple second-level offices were needed, as illustrated in Fig. 6.1 (c). Eventually, the hierarchy grew to five levels.

While there have been improvements, the basic Bell System model has remained essentially intact for over 100 years. The following description is highly simplified but gives the essential flavor nevertheless.Each telephone has two copper wires coming out of it that go directly to the telephone company's nearest end office (also called a local central office). The distance is typically 1 to 10 km, being shorter in cities than in rural areas. In the United States alone there are about 22,000 end offices. The two-wire connections between each subscriber's telephone and the end office are known in the trade as the local loop.

If a subscriber attached to a given end office calls another subscriber attachedto the same end office, the switching mechanism within the office sets up a direct electrical connection between the two local loops. This connection remains intact for the duration of the call.If the called telephone is attached to another end office, a different procedure has to be used. Each end office has a number of outgoing lines to one or more nearby switching centers, called toll offices (or, if they are within the same localarea, tandem offices). These lines are called toll connecting trunks. The number of different kinds of switching centers and their topology varies from country to country depending on the country's telephone density.

If both the caller's and callee's end offices happen to have a toll connecting trunk to the same toll office,the connection may be established within the toll office. A telephone network consisting only of telephones (the small dots), end offices (the large dots), and tolloffices (the squares) is shown in Fig. 6.1 (c).

If the caller and callee do not have a toll office in common, a path will have to be established between two toll offices. The toll offices communicate with each other via high-bandwidth intertoll trunks (also called interoffice trunks). Priorto the 1984 breakup of AT&T, the U.S. telephone system used hierarchical routing to find a path, going to higher levels of the hierarchy until there

was a switchingoffice in common. This was then replaced with more flexible, nonhierarchical routing. Figure 6.2 shows how a long-distance connection might be routed.

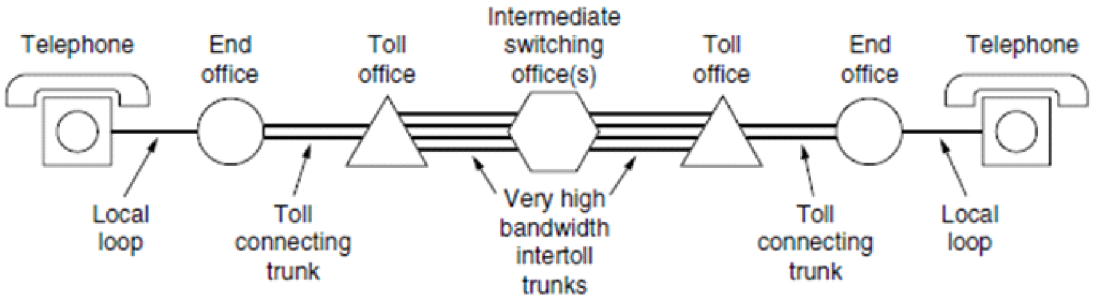


Figure 6.2 A typical circuit route for a long-distance call

A variety of transmission media are used for telecommunication. Unlike modern office buildings, where the wiring is commonly Category 5, local loops to homes mostly consist of Category 3 twisted pairs, with fiber just starting to appear. Between switching offices, coaxial cables, microwaves, and especially fiber optics are widely used.

In the past, transmission throughout the telephone system was analog, with the actual voice signal being transmitted as an electrical voltage from source to destination. With the advent of fiber optics, digital electronics, and computers, allthe trunks and switches are now digital, leaving the local loop as the last piece of analog technology in the system. Digital transmission is preferred because it isnot necessary to accurately reproduce an analog waveform after it has passed through many amplifiers on a long call. Being able to correctly distinguish a 0 from a 1 is enough. This property makes digital transmission more reliable than analog. It is also cheaper and easier to maintain.In summary, the telephone system consists of three major components:

1. Local loops (analog twisted pairs going to houses and businesses).
2. Trunks (digital fiber optic links connecting the switching offices).
3. Switching offices (where calls are moved from one trunk to another).

6.4 The Local Loop: Modems, ADSL and Fiber

It is now time to start our detailed study of how the telephone system works.Let us begin with the part that most people are familiar with: the two-wire localloop coming from a telephone

company end office into houses. It has carried analog information for over 100 years and is likely to continue doing so for some years to come, due to the high cost of converting todigital.

Much effort has been devoted to squeezing data networking out of the copper local loops that are already deployed. Telephone modems send digital data between computers over the narrow channel the telephone network provides for a voice call. They were once widely used, but have been largely displaced by broadband technologies such as ADSL that reuse the local loop to send digital data from a customer to the end office, where they are siphoned off to the Internet.Both modems and ADSL must deal with the limitations of old local loops: relativelynarrow bandwidth, attenuation and distortion of signals, and susceptibility to electrical noise such as crosstalk.

In some places, the local loop has been modernized by installing optical fiber to (or very close to) the home. Fiber is the way of the future. These installations support computer networks from the ground up, with the local loop having ample band width for data services. The limiting factor is what people will pay, not the physics of the local loop.

6.4.1 Telephone Modems

To send bits over the local loop, or any other physical channel for that matter,they must be converted to analog signals that can be transmitted over the channel.This conversion is accomplished using the methods for digital modulation. At the other end of the channel, the analog signal is converted back to bits.

A device that converts between a stream of digital bits and an analog signal that represents the bits is called a modem, which is short for "modulator demodulator."Modems come in many varieties: telephone modems, DSL modems, cable modems, wireless modems, etc. The modem may be built into the computer(which is now common for telephone modems) or be a separate box (which iscommon for DSL and cable modems). Logically, the modem is inserted betweenthe (digital) computer and the (analog) telephone system, as seen in Fig. 6.3.

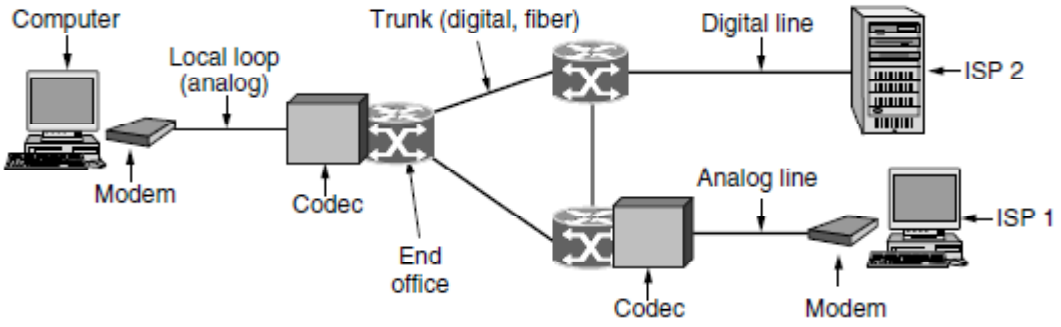


Figure 6.3. The use of both analog and digital transmission for a computer-to-computer call. Conversion is done by the modems and codecs.

Telephone modems are used to send bits between two computers over avoice-grade telephone line, in place of the conversation that usually fills the line.The main difficulty in doing so is that a voice-grade telephone line is limited to3100 Hz, about what is sufficient to carry a conversation. This bandwidth is more than four orders of magnitude less than the bandwidth that is used for Ethernet or 802.11 (WiFi). Unsurprisingly, the data rates of telephone modems are also four orders of magnitude less than that of Ethernet and 802.11.

A telephone channel is carried inside the telephone system as digital samples. Each telephone channel is 4000 Hz wide when the guard bands are included. The number of samples per second needed to reconstruct it is thus 8000. The number of bits per sample in the U.S. is 8, one ofwhich may be used for control purposes, allowing 56,000 bits/sec of user data. In Europe, all 8 bits are available to users, so 64,000-bit/sec modems could have been used, but to get international agreement on a standard, 56,000 was chosen.

The end result is the V.90 and V.92 modem standards. They provide for a56-kbps downstream channel (ISP to user) and a 33.6-kbps and 48-kbps upstream channel (user to ISP), respectively. The asymmetry is because there is usually more data transported from the ISP to the user than the other way. It also means that more of the limited bandwidth can be allocated to the downstream channel to increase the chances of it actually working at 56 kbps.

6.4.2 Digital Subscriber Lines

When the telephone industry finally got to 56 kbps, it patted itself on the back for a job well done. Meanwhile, the cable TV industry was offering speeds up to10 Mbps on shared

cables. As Internet access became an increasingly important part of their business, the telephone companies (LECs) began to realize they needed a more competitive product. Their answer was to offer new digital services over the local loop.

Initially, there were many overlapping high-speed offerings, all under the generalname of xDSL(Digital Subscriber Line), for various x. Services with more bandwidth than standard telephone service are sometimes called broadband, although the term really is more of a marketing concept than a specific technical concept. The most popular of these services,ADSL (Asymmetric DSL).

The reason that modems are so slow is that telephones were invented for carryingthe human voice and the entire system has been carefully optimized for this purpose. Data have always been stepchildren. At the point where each local loop terminates in the end office, the wire runs through a filter that attenuates all frequencies below 300 Hz and above 3400 Hz. The cutoff is not sharp-300 Hz and3400 Hz are the 3-dB points-so the bandwidth is usually quoted as 4000 Hz eventhough the distance between the 3 dB points is 3100 Hz. Data on the wire are thus also restricted to this narrow band.

The trick that makes xDSL work is that when a customer subscribes to it, the incoming line is connected to a different kind of switch, one that does not have this filter, thus making the entire capacity of the local loop available. The limiting factor then becomes the physics of the local loop, which supports roughly 1 MHz,not the artificial 3100 Hz bandwidth created by the filter.

Unfortunately, the capacity of the local loop falls rather quickly with distance from the end office as the signal is increasingly degraded along the wire. It also depends on the thickness and general quality of the twisted pair. A plot of the potential bandwidth as a function of distance is given in Fig. 6.4. This figure assumes that all the other factors are optimal (new wires, modest bundles, etc.).

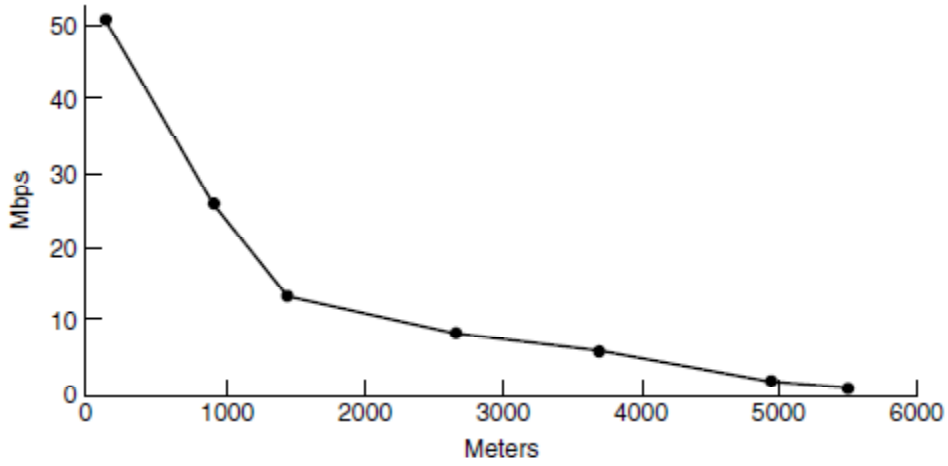


Figure 6.4 Bandwidth versus distance over Category 3 UTP for DSL

The implication of this figure creates a problem for the telephone company.When it picks a speed to offer, it is simultaneously picking a radius from its endoffices beyond which the service cannot be offered. This means that when distant customers try to sign up for the service, they could not get the service because they live 100 meters too far from the nearest end office. The lower the chosen speed is, the larger the radius and the more customers are covered. But the lower the speed, the less attractive the service is and the fewer the people who will be willing to pay for it.This is where business meets technology.

The xDSL services have all been designed with certain goals in mind. First,the services must work over the existing Category 3 twisted pair local loops. Second,they must not affect customers' existing telephones and fax machines. Third,they must be much faster than 56 kbps. Fourth, they should be always on, withjust a monthly charge and no per-minute charge.

To meet the technical goals, the available 1.1 MHz spectrum on the local loopis divided into 256 independent channels of 4312.5 Hz each. This arrangement is shown in Fig. 6.5. The OFDM scheme, which we saw in the previous section, isused to send data over these channels, though it is often called DMT (Discrete MultiTone) in the context of ADSL. Channel 0 is used for POTS (Plain Old Telephone Service). Channels 1-5 are not used, to keep the voice and data signals from interfering with each other. Of the remaining 250 channels, one is used for upstream control and one is used for downstream control. The rest are available for user data.

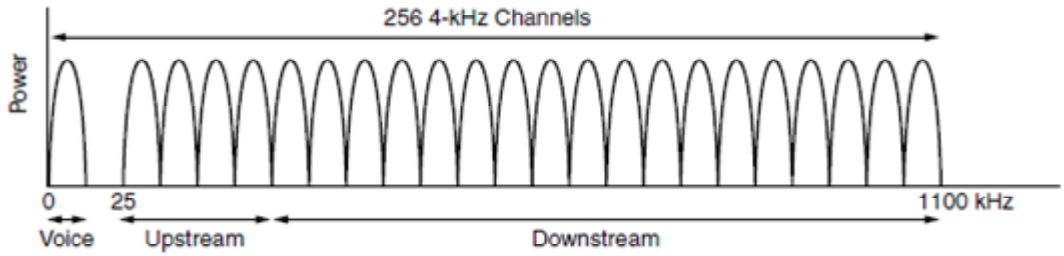


Figure 6.5 Operation of ADSL using discrete multitone modulation

In principle, each of the remaining channels can be used for a full-duplex datastream, but harmonics, crosstalk, and other effects keep practical systems well below the theoretical limit. It is up to the provider to determine how many channels are used for upstream and how many for downstream. A 50/50 mix of upstream and downstream is technically possible, but most providers allocate something like 80-90% of the bandwidth to the downstream channel since most users download more data than they upload. This choice gives rise to the "A" in ADSL. A common split is 32 channels for upstream and the rest downstream. It is also possible to have a few of the highest upstream channels be bidirectional for increased bandwidth, although making this optimization requires adding a specialcircuit to cancel echoes.

Within each channel, QAM (Quadrature Amplitude Modulation) is used at a rate of roughly 4000 symbols/sec. The line quality in each channel is constantly monitored and thedata rate is adjusted by using a larger or smaller constellation. Different channels may have different data rates, with up to 15 bits persymbol sent on a channel with a high SNR, and down to 2, 1, or no bits per symbol sent on a channel with a low SNR depending on the standard.

A typical ADSL arrangement is shown in Fig. 6.6. In this scheme, a telephonecompany technician must install a NID (Network Interface Device) on the customer's premises. This small plastic box marks the end of the telephone company's property and the start of the customer's property. Close to the NID (or sometimes combined with it) is a splitter, an analog filter that separates the0-4000-Hz band used by POTS from the data. The POTS signal is routed to the existing telephone or fax machine. The data signal is routed to an ADSL

modem,which uses digital signal processing to implement OFDM. Since most ADSL modems are external, the computer must be connected to them at high speed.Usually, this is done using Ethernet, a USB cable, or 802.11.

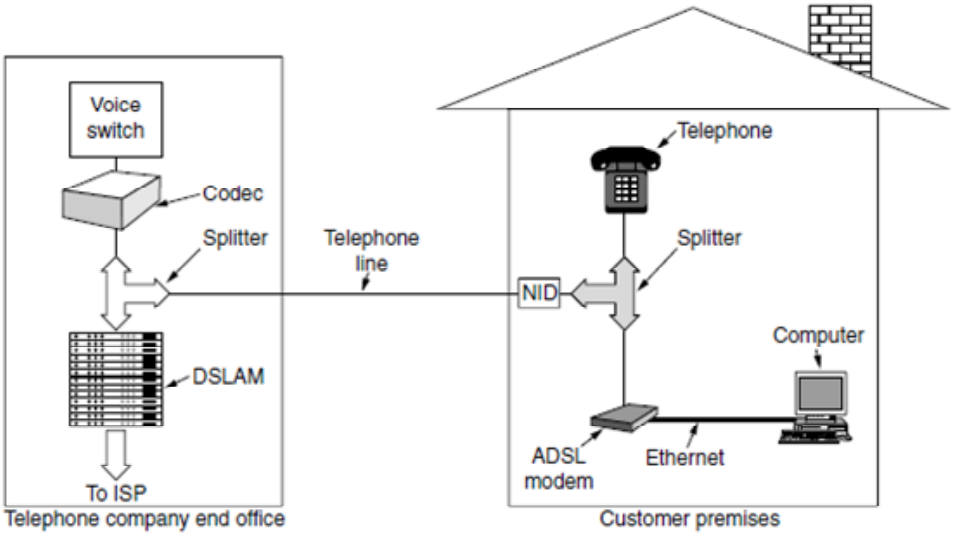


Figure 6.6A typical ADSL equipment configuration

At the other end of the wire, on the end office side, a corresponding splitter is installed. Here, the voice portion of the signal is filtered out and sent to the normal voice switch. The signal above 26 kHz is routed to a new kind of device called a DSLAM (Digital Subscriber Line Access Multiplexer), which contains the same kind of digital signal processor as the ADSL modem. Once the bits have been recovered from the signal, packets are formed and sent off to the ISP.

This complete separation between the voice system and ADSL makes it relatively easy for a telephone company to deploy ADSL. All that is needed is buying a DSLAM and splitter and attaching the ADSL subscribers to the splitter.Other high-bandwidth services (e.g., ISDN) require much greater changes to the existing switching equipment.

One disadvantage of the design of Fig. 6.6 is the need for a NID (Network Interface Device) and splitter on the customer's premises. Installing these can only be done by a telephone company technician, necessitating an expensive "truck roll" (i.e., sending a technicianto the customer's premises). Therefore, an alternative, splitterless design,informally called G.lite, has also been standardized. It is the same as Fig. 6.6 but without the customer's splitter. The

existing telephone line is used as is. The only difference is that a microfilter has to be inserted into each telephone jack between the telephone or ADSL modem and the wire. The microfilter for the telephone is a low-pass filter eliminating frequencies above 3400 Hz; the microfilterfor the ADSL modem is a high-pass filter eliminating frequencies below 26kHz. However, this system is not as reliable as having a splitter, so G.lite can beused only up to 1.5 Mbps (versus 8 Mbps for ADSL with a splitter).

6.4.3 Fiber To The Home

Deployed copper local loops limit the performance of ADSL and telephone modems. To let them provide faster and better network services, telephone companies are upgrading local loops at every opportunity by installing optical fiber all the way to houses and offices. The result is called FttH(Fiber To The Home).While FttH technology has been available for some time, deployments only began to take off in 2005 with growth in the demand for high-speed Internet from customers used to DSL and cable who wanted to download movies. Around 4% ofU.S. houses are now connected to FttH with Internet access speeds of up to 100Mbps.

Several variations of the form ''FttX'' (where X stands for the basement, curb,or neighborhood) exist. They are used to note that the fiber deployment may reach close to the house. In this case, copper (twisted pair or coaxial cable) provides fast enough speeds over the last short distance. The choice of how far to lay the fiber is an economic one, balancing cost with expected revenue. In any case,the point is that optical fiber has crossed the traditional barrier of the ''last mile.''We will focus on FttH in our discussion.

Like the copper wires before it, the fiber local loop is passive. This means no powered equipment is required to amplify or otherwise process signals. The fiber simply carries signals between the home and the end office. This in turn reduces cost and improves reliability.

Usually, the fibers from the houses are joined together so that only a single fiber reaches the end office per group of up to 100 houses. In the downstream direction,optical splitters divide the signal from the end office so that it reaches all the houses. Encryption is needed for security if only one house should be able to decode the signal. In the upstream direction, optical combiners merge the signals from the houses into a single signal that is received at the end office.

This architecture is called a PON (Passive Optical Network), and it is shown in Fig. 6.7. It is common to use one wavelength shared between all the houses for downstream transmission, and another wavelength for upstream transmission.Even with the splitting, the tremendous bandwidth and low attenuation of fiber mean that PONs can provide high rates to users over distances of up to 20km. The actual data rates and other details depend on the type of PON. Two kinds are common. GPONs (Gigabit-capable PONs) come from the world of telecommunications,so they are defined by an ITU standard. EPONs (Ethernet PONs)are more in tune with the world of networking, so they are defined by an IEEE standard. Both run at around a gigabit and can carry traffic for different services,including Internet, video, and voice. For example, GPONs provide 2.4 Gbps downstream and 1.2 or 2.4 Gbps upstream.

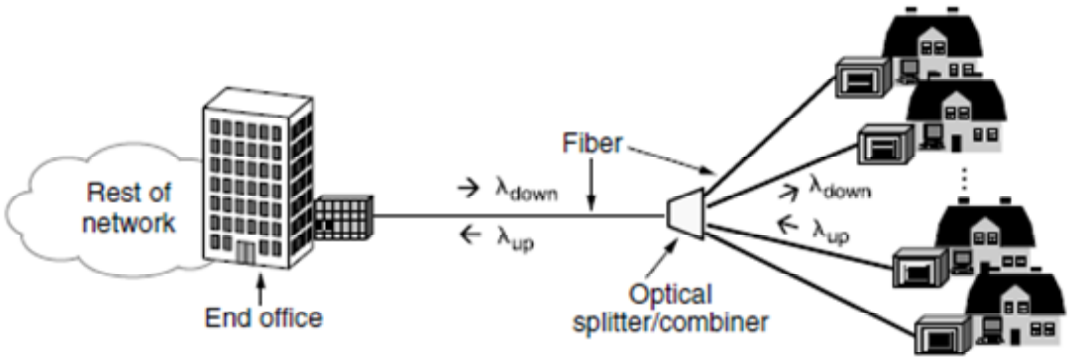


Figure 6.7 Passive optical network for Fiber To The Home

Some protocol is needed to share the capacity of the single fiber at the endoffice between the different houses. The downstream direction is easy. The endoffice can send messages to each different house in whatever order it likes. In the upstream direction, however, messages from different houses cannot be sent at the same time, or different signals would collide. The houses also cannot hear each other's transmissions so they cannot listen before transmitting. The solution is that equipment at the houses requests and is granted time slots to use by equipmentin the end office. For this to work, there is a ranging process to adjust the transmission times from the houses so that all the signals received at the endoffice are synchronized.

6.5 Trunks and Multiplexing

Trunks in the telephone network are not only much faster than the local loops,they are different in two other respects. The core of the telephone network carries conversion at the end office to digital form for transmission over the long-haul trunks. The trunks carry thousands, even millions, of calls simultaneously.This sharing is important for achieving economies of scale, since it costs essentially the same amount of money to install and maintain a high-bandwidth trunk asa low-bandwidth trunk between two switching offices. It is accomplished with versions of TDM (Time Division Multiplexing) and FDM (Frequency Division Multiplexing)

Below we will briefly examine how voice signals are digitized so that they can be transported by the telephone network. After that, we will see how TDM is used to carry bits on trunks, including the TDM system used for fiber optics(SONET). Then we will turn to FDM as it is applied to fiber optics, which is called wavelength division multiplexing.

6.5.1 Digitizing Voice Signals

Early in the development of the telephone network, the core handled voice calls as analog information. FDM techniques were used for many years to multiplex 4000-Hz voice channels (comprised of 3100 Hz plus guard bands) into larger and larger units. For example, 12 calls in the 60 kHz-to-108 kHz band is known as a group and five groups (a total of 60 calls) are known as a supergroup, and so on. These FDM methods are still used over some copper wires and microwave channels. However, FDM requires analog circuitry and is not amenable to beingdone by a computer. In contrast, TDM can be handled entirely by digital electronics,so it has become far more widespread in recent years. Since TDM can only be used for digital data and the local loops produce analog signals, a conversionis needed from analog to digital in the end office, where all the individual local loops come together to be combined onto outgoing trunks.

The analog signals are digitized in the end office by a device called a codec(short for "coder-decoder"). The codec makes 8000 samples per second (125?sec/sample) because the Nyquist theorem says that this is sufficient to capture all the information from the 4-kHz telephone channel bandwidth. At a lower sampling rate, information would be lost; at a higher one, no extra information would be gained. Each sample of the amplitude of the signal is quantized to an 8-bit number.

This technique is called PCM (Pulse Code Modulation). It forms the heart of the modern telephone system. As a consequence, virtually all time intervals within the telephone system are multiples of 125 μsec. The standard uncompressed data rate for a voice-grade telephone call is thus 8 bits every 125μsec, or 64 kbps.

At the other end of the call, an analog signal is recreated from the quantized samples by playing them out (and smoothing them) over time. It will not be exactly the same as the original analog signal, even though we sampled at the Nyquist rate, because the samples were quantized. To reduce the error due to quantization, the quantization levels are unevenly spaced. A logarithmic scale isused that gives relatively more bits to smaller signal amplitudes and relatively fewer bits to large signal amplitudes. In this way the error is proportional to the signal amplitude.

6.5.2 Time Division Multiplexing

TDM based on PCM is used to carry multiple voice calls over trunks by sending a sample from each call every 125 μsec. When digital transmission began emerging as a feasible technology, ITU (then called CCITT) was unable to reach agreement on an international standard for PCM. Consequently, a variety of incompatible schemes are now in use in different countries around the world.The method used in North America and Japan is the T1 carrier, depicted inFig. 6.8. The T1 carrier consists of 24 voice channels multiplexed together.Each of the 24 channels, in turn, gets to insert 8 bits into the output stream.

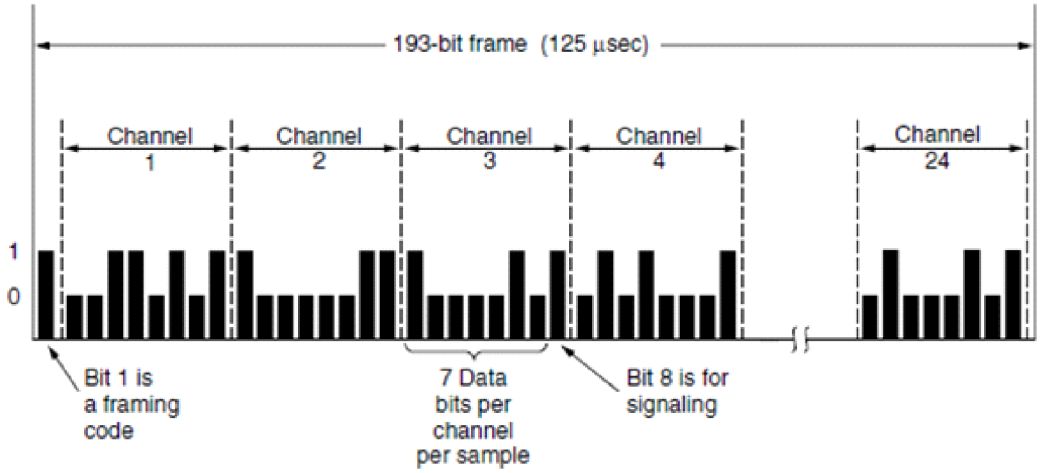


Figure 6.8. The T1 carrier (1.544 Mbps)

A frame consists of 24 × 8 = 192 bits plus one extra bit for control purposes, yielding 193 bits every 125 ?sec. This gives a gross data rate of 1.544 Mbps, of which 8 kbps is for signaling. The 193rd bit is used for frame synchronization and signaling. In one variation, the 193rd bit is used across a group of 24 frames called an extended superframe. Six of the bits, in the 4th, 8th, 12th, 16th, 20th, and 24th positions, take on the alternating pattern 001011 . . . . Normally, the receiver keeps checking for this pattern to make sure that it has not lost synchronization.Six more bits are used to send an error check code to help the receiver confirm that it is synchronized. If it does get out of sync, the receiver can scan for the pattern and validate the error check code to get resynchronized. The remaining 12 bits are used for control information for operating and maintaining the network,such as performance reporting from the remote end.

The T1 format has several variations. The earlier versions sent signalling information in-band, meaning in the same channel as the data, by using some of the data bits. This design is one form of channel-associated signaling, because each channel has its own private signaling subchannel. In one arrangement, the least significant bit out of an 8-bit sample on each channel is used in every sixth frame. It has the colorful name of robbed-bit signaling. The idea is that a few stolen bits will not matter for voice calls. No one will hear the difference.

If older versions of T1 are used to carry data, only 7 of 8 bits, or56 kbps can be used in each of the 24 channels. Instead, newer versions of T1 provide clear channels in which all of the bits may be used to send data. Clear channels are what businesses who lease a T1 line want when they send data across the telephone network in place of voice samples. Signaling for any voice calls is then handled out-of-band, meaning in a separate channel from the data. Often,the signaling is done with common-channel signaling in which there is a shared signaling channel. One of the 24 channels may be used for this purpose.

Time division multiplexing allows multiple T1 carriers to be multiplexed into higher-order carriers. Figure 6.9 shows how this can be done. At the left we see four T1 channels being multiplexed into one T2 channel. The multiplexing at T2 and above is done bit for bit, rather than byte for byte with the 24 voice channels that make up a T1 frame. Four T1 streams at 1.544 Mbps should generate 6.176 Mbps, but T2 is actually 6.312 Mbps. The extra bits are used for framing and recoveryin case the carrier slips. T1 and T3 are widely used by customers, whereasT2 and T4 are only used within the telephone system itself, so they are not well known.

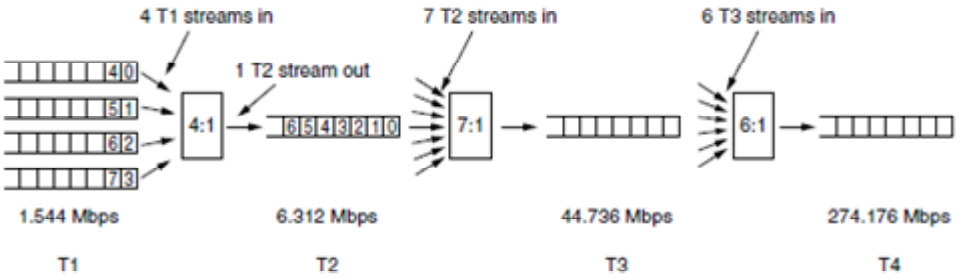


Figure 6.9 Multiplexing T1 streams into higher carriers

At the next level, seven T2 streams are combined bitwise to form a T3 stream. Then six T3 streams are joined to form a T4 stream. At each step a small amount of overhead is added for framing and recovery in case the synchronization between sender and receiver is lost.

6.5.3 SONET/DH

In the early days of fiber optics, every telephone company had its own proprietary optical TDM system. After AT&T was broken up in 1984, local telephone companies had to connect to multiple long-distance carriers, all with different optical TDM systems, so the need for standardization became obvious. In 1985, Bellcore, the RBOC's research arm, began working on a standard, called SONET (Synchronous Optical NETwork).

Later, ITU joined the effort, which resulted in a SONET standard and a set of parallel ITU recommendations (G.707, G.708, and G.709) in 1989. The ITU recommendations are called SDH (Synchronous Digital Hierarchy) but differ from SONET only in minor ways. Virtually all the long-distance telephone traffic in the United States, and much of it elsewhere, now uses trunks running SONET in the physical layer.

The SONET design had four major goals. First and foremost, SONET had tomake it possible for different carriers to interwork. Achieving this goal required defining a common signaling standard with respect to wavelength, timing, framing structure, and other issues.

Second, some means was needed to unify the U.S., European, and Japanese digital systems, all of which were based on 64-kbps PCM channels but combined them in different (and incompatible) ways.

Third, SONET had to provide a way to multiplex multiple digital channels.At the time SONET was devised, the highest-speed digital carrier actually used widely in the United States was T3, at 44.736 Mbps. T4 was defined, but not used much, and nothing was even defined above T4 speed. Part of SONET's mission was to continue the hierarchy to gigabits/sec and beyond. A standard way to multiplex slower channels into one SONET channel was also needed.

Fourth, SONET had to provide support for operations, administration, and maintenance (OAM), which are needed to manage the network. Previous systems did not do this very well. The basic SONET frame is a block of 810 bytes put out every 125 μsec.Since SONET is synchronous, frames are emitted whether or not there are any useful data to send. Having 8000 frames/sec exactly matches the sampling rate of the PCM channels used in all digital telephony systems.

The 810-byte SONET frames are best described as a rectangle of bytes, 90 columns wide by 9 rows high. Thus, 8 × 810 = 6480 bits are transmitted 8000 times per second, for a gross data rate of 51.84 Mbps. This layout is the basic SONET channel, called STS-1 (Synchronous Transport Signal-1). All SONET trunks are multiples of STS-1.

The first three columns of each frame are reserved for system management information, as illustrated in Fig. 6.10. In this block, the first three rows contain the section overhead; the next six contain the line overhead. The section overhead is generated and checked at the start and end of each section, whereas the line overhead is generated and checked at the start and end of each line.

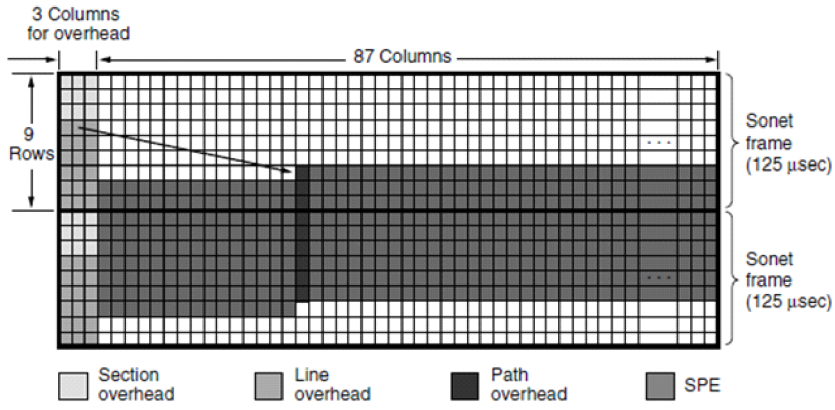


Figure 6.10 Two back-to-back SONET frames

A SONET transmitter sends back-to-back 810-byte frames, without gaps between them, even when there are no data (in which case it sends dummy data).From the receiver's point of view, all it sees is a continuous bit stream, so how does it know where each frame begins? The answer is that the first 2 bytes ofeach frame contain a fixed pattern that the receiver searches for. If it finds this pattern in the same place in a large number of consecutive frames, it assumes that it is in sync with the sender. In theory, a user could insert this pattern into the payload in a regular way, but in practice it cannot be done due to the multiplexing of multiple users into the same frame and other reasons.

The remaining 87 columns of each frame hold 87 × 9 × 8 × 8000 = 50.112 Mbps of user data. This user data could be voice samples, T1 and other carriers swallowed whole, or packets. SONET is simply a convenient container for transporting bits. The SPE (Synchronous Payload Envelope), which carries the user data does not always begin in row 1, column 4. The SPE can begin anywhere within the frame. A pointer to the first byte is contained in the first row of the line overhead. The first column of the SPE is the path overhead (i.e., the header forthe end-to-end path sublayer protocol).

The ability to allow the SPE to begin anywhere within the SONET frame and even to span two frames, as shown in Fig. 6.10, gives added flexibility to the system.For example, if a payload arrives at the source while a dummy SONET frame is being constructed, it can be inserted into the current frame instead of being held until the start of the next one.

6.5.4 Wavelength Division Multiplexing

A form of frequency division multiplexing is used as well as TDM to harness the tremendous bandwidth of fiber optic channels. It is called WDM (Wavelength Division Multiplexing). The basic principle of WDM on fibers is depicted in Fig. 6.11. Here four fibers come together at an optical combiner, each with its energy present at a different wavelength. The four beams are combined onto a single shared fiber for transmission to a distant destination. At the far end,the beam is split up over as many fibers as there were on the input side. Each output fiber contains a short, specially constructed core that filters out all but one wavelength. The resulting signals can be routed to their destination or recombined in different ways for additional multiplexed transport.

There is really nothing new here. This way of operating is just frequency division multiplexing at very high frequencies, with the term WDM owing to the description of fiber optic channels by their wavelength or "color" rather than frequency.As long as each channel has its own frequency (i.e., wavelength) range and all the ranges are disjoint, they can be multiplexed together on the long-haul fiber. The only difference with electrical FDM is that an optical system using adiffraction grating is completely passive and thus highly reliable.

The reason WDM is popular is that the energy on a single channel is typically only a few gigahertz wide because that is the current limit of how fast we can convert between electrical and optical signals. By running many channels in parallel on different wavelengths, the aggregate bandwidth is increased linearly with the number of channels. Since the bandwidth of a single fiber band is about 25,000 GHz, there is theoretically room for 2500 10-Gbps channels even at 1 bit/Hz (and higher rates are also possible).

WDM technology has been progressing at a rate that puts computer technologyto shame. WDM was invented around 1990. The first commercial systems had eight channels of 2.5 Gbps per channel. By 1998, systems with 40 channels of 2.5 Gbps were on the market. By 2006, there were products with 192 channels of 10 Gbps and 64 channels of 40 Gbps, capable of moving up to 2.56 Tbps. This bandwidth is enough to transmit 80 full-length DVD movies per second. The channels are also packed tightly on the fiber, with 200, 100, or as little as 50 GHzof separation. Technology demonstrations by companies after bragging rights have shown 10 times this capacity in the lab, but going from the lab to the field usually takes at least a few years. When the number of channels is very large and the wavelengths are spaced close together, the system is referred to as DWDM(Dense WDM).

One of the drivers of WDM technology is the development of all-optical components.Previously, every 100 km it was necessary to split up all the channels and convert each one to an electrical signal for amplification separately before re-converting them to optical signals and combining them. Nowadays, all-optical amplifiers can regenerate the entire signal once every 1000 km without the need for multiple opto-electrical conversions.

In the example of Fig. 6.11, we have a fixed-wavelength system. Bits from input fiber 1 go to output fiber 3, bits from input fiber 2 go to output fiber 1, etc.However, it is also possible to

build WDM systems that are switched in the optical domain. In such a device, the output filters are tunable using Fabry-Perot or Mach-Zehnder interferometers. These devices allow the selected frequencies to be changed dynamically by a control computer. This ability provides a large amount of flexibility to provision many different wavelength paths through the telephone network from a fixed set of fibers. For more information about optical net works and WDM, see Ramaswami et al. (2009).

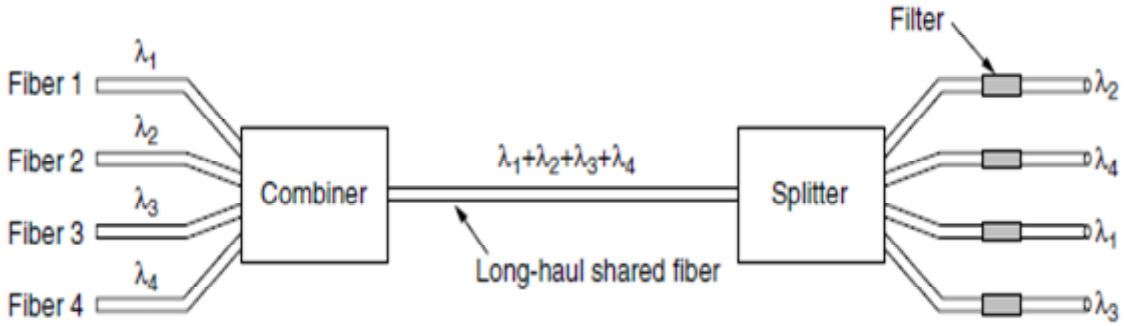


Figure 6.11. Wavelength division multiplexing

6.6 Switching

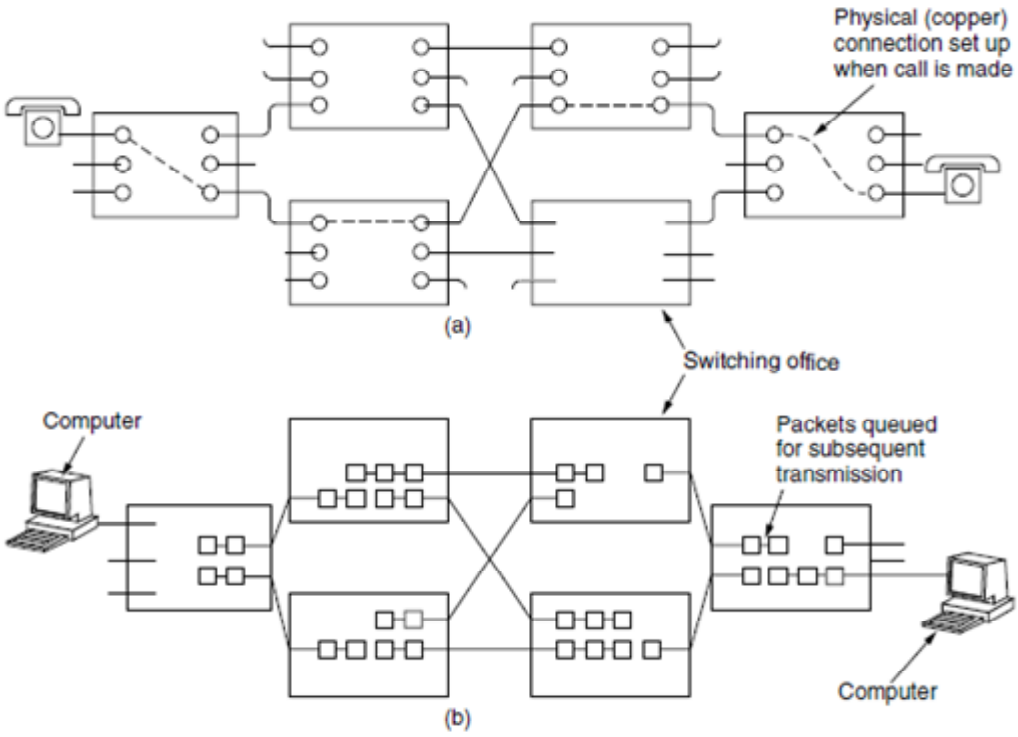
From the point of view of the average telephone engineer, the phone system is divided into two principal parts: outside plant (the local loops and trunks, since they are physically outside the switching offices) and inside plant (the switches,which are inside the switching offices). We have just looked at the outside plant.Now it is time to examine the inside plant.

Two different switching techniques are used by the network nowadays: circuit switching and packet switching. The traditional telephone system is based on circuit switching, but packet switching is beginning to make in roads with the rise of voice over IP technology. We will go into circuit switching in some detail and contrast it with packet switching.

6.6.1Circuit Switching

Conceptually, when you or your computer places a telephone call, the switching equipment within the telephone system seeks out a physical path all the way from your telephone to the receiver's telephone. This technique is called circuit switching. It is shown schematically in

Fig. 6.12(a). Each of the six rectangles represents a carrier switching office (end office, toll office, etc.). In this example,each office has three incoming lines and three outgoing lines. When a call passes through a switching office, a physical connection is (conceptually) established between the line on which the call came in and one of the output lines, as shown by the dotted



lines.

Figure 6.12(a) Circuit switching. (b) Packet switching.

In the early days of the telephone, the connection was made by the operator plugging a jumper cable into the input and output sockets. In fact, a surprising little story is associated with the invention of automatic circuit switching equipment.It was invented by a 19th-century Missouri undertaker named Almon B. Strowger shortly after the telephone was invented, when someone died, one of the survivors would call the town operator and say "Please connect me to an undertaker." Unfortunately for Mr. Strowger, there were two undertakers in his town, and theother one's wife was the town telephone operator. He quickly saw that either hewas going to have to invent automatic telephone switching equipment or he wasgoing to go out of business. He chose the first option. For nearly 100 years, thecircuit-switching equipment used worldwide was known as Strowger gear.

The model shown in Fig. 6.12(a) is highly simplified, of course, because parts of the physical path between the two telephones may, in fact, be microwave or fiber links onto which thousands of calls are multiplexed. Nevertheless, the basic idea is valid: once a call has been set up, a dedicated path between both ends exists and will continue to exist until the call is finished.

An important property of circuit switching is the need to set up an end-to-end path before any data can be sent. The elapsed time between the end of dialing and the start of ringing can easily be 10 sec, more on long-distance or international calls. During this time interval, the telephone system is hunting for a path, as shown in Fig. 6.13(a). Note that before data transmission can even begin, the call request signal must propagate all the way to the destination and be acknowledged.For many computer applications (e.g., point-of-sale credit verification), long setuptimes are undesirable.

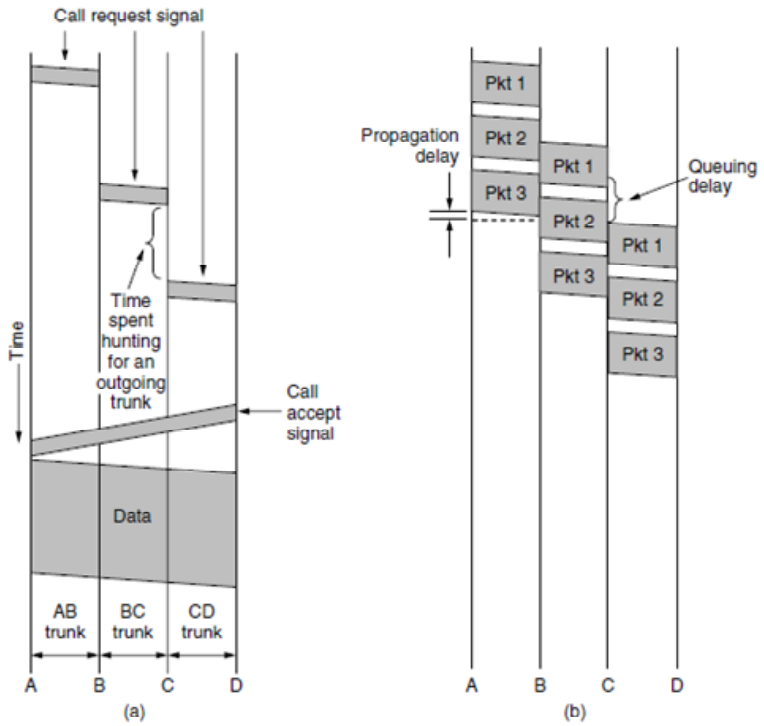


Figure 6.13 Timing of events in (a) circuit switching, (b) packet switching

As a consequence of the reserved path between the calling parties, once the setup has been completed, the only delay for data is the propagation time for the electromagnetic signal,

about 5 msec per 1000 km. Also as a consequence of the established path, there is no danger of congestion-that is, once the call has been put through, you never get busy signals. Of course, you might get one before the connection has been established due to lack of switching or trunk capacity.

6.6.2 Packet Switching

The alternative to circuit switching is packet switching, shown in Fig. 6.12(b). With this technology, packets are sent as soon as they are available. There is no need to set up a dedicated path in advance, unlike with circuit switching. It is up to routers to use store-and-forward transmission to send each packet on its way to the destination on its own. This procedure isunlike circuit switching, in which the result of the connection setup is the reservation of bandwidth all the way from the sender to the receiver. All data on the circuit follows this path. Among other properties, having all the data follow the same path means that it cannot arrive out of order. With packet switching there is no fixed path, so different packets can follow different paths, depending on network conditions at the time they are sent, and they may arrive out of order.

Packet-switching networks place a tight upper limit on the size of packets.This ensures that no user can monopolize any transmission line for very long (e.g.,many milliseconds), so that packet-switched networks can handle interactive traffic.It also reduces delay since the first packet of a long message can be forwarded before the second one has fully arrived. However, the store-and-forward delay of accumulating a packet in the router's memory before it is sent on to thenext router exceeds that of circuit switching. With circuit switching, the bits just flow through the wire continuously.

Packet and circuit switching also differ in other ways. Because no bandwidth is reserved with packet switching, packets may have to wait to be forwarded.This introduces queuing delay and congestion if many packets are sent at the same time. On the other hand, there is no danger of getting a busy signal and being unable to use the network. Thus, congestion occurs at different times with circuit switching (at setup time) and packet switching (when packets are sent).

If a circuit has been reserved for a particular user and there is no traffic, its bandwidth is wasted. It cannot be used for other traffic. Packet switching does not waste bandwidth and thus

is more efficient from a system perspective. Understanding this trade-off is crucial for comprehending the difference between circuit switching and packet switching. The trade-off is between guaranteed service and wasting resources versus not guaranteeing service and not wasting resources.

Packet switching is more fault tolerant than circuit switching. In fact, that is why it was invented. If a switch goes down, all of the circuits using it are terminated and no more traffic can be sent on any of them. With packet switching,packets can be routed around dead switches.

A final difference between circuit and packet switching is the charging algorithm.With circuit switching, charging has historically been based on distance and time. For mobile phones, distance usually does not play a role, except for international calls, and time plays only a coarse role (e.g., a calling plan with 2000 free minutes costs more than one with 1000 free minutes and sometimes nights or weekends are cheap). With packet switching, connect time is not an issue, but the volume of traffic is. For home users, ISPs usually charge a flat monthly rate because it is less work for them and their customers can understand this model, but backbone carriers charge regional networks based on the volume of their traffic.

The differences are summarized in Fig. 6.14. Traditionally, telephone networks have used circuit switching to provide high-quality telephone calls, and computer networks have used packet switching for simplicity and efficiency.However, there are notable exceptions. Some older computer networks have been circuit switched under the covers (e.g., X.25) and some newer telephone networks use packet switching with voice over IP technology. This looks just like a standard telephone call on the outside to users, but inside the network packets of voice data are switched. This approach has let upstarts market cheap international calls via calling cards, though perhaps with lower call quality than the incumbents.

Item	Circuit switched	Packet switched
Call setup	Required	Not needed
Dedicated physical path	Yes	No
Each packet follows the same route	Yes	No
Packets arrive in order	Yes	No
Is a switch crash fatal	Yes	No
Bandwidth available	Fixed	Dynamic
Time of possible congestion	At setup time	On every packet
Potentially wasted bandwidth	Yes	No
Store-and-forward transmission	No	Yes
Charging	Per minute	Per packet

Figure 6.14 A comparison of circuit-switched and packet-switched networks

6.7 Summary

A key element in most wide area networks is the telephone system. Its main components are the local loops, trunks, and switches. ADSL offers speeds up to40 Mbps over the local loop by dividing it into many subcarriers that run in parallel.This far exceeds the rates of telephone modems. PONs bring fiber to the home for even greater access rates than ADSL.

Trunks carry digital information. They are multiplexed with WDM to provision many high capacity links over individual fibers, as well as with TDM to share each high rate link between users. Both circuit switching and packet switching are important.

6.8 Model Questions

1. Describe the structure of the Telephone system.
2. Explain in detail about multiplexing and it's types.
3. Illustrate Switching.
4. Write notes on circuit switching.
5. Discuss on packet switching.

LESSON-7
THE DATA LINK LAYER

Structure

7.1 Introduction

In this lesson, we will study the design principles for the second layer in our model, the data link layer. This study deals with algorithms for achieving reliable,efficient communication of whole units of information called frames (rather than individual bits, as in the physical layer) between two adjacent machines.By adjacent, we mean that the two machines are connected by a communication channel that acts conceptually like a wire (e.g., a coaxial cable, telephone line, or wireless channel). The essential property of a channel that makes it ''wire-like''is that the bits are delivered in exactly the same order in which they are sent.

At first you might think this problem is so trivial that there is nothing to study-machine A just puts the bits on the wire, and machine B just takes them off. Unfortunately, communication channels make errors occasionally. Furthermore,they have only a finite data rate, and there is a nonzero propagation delay between the time a bit is sent and the time it is received. These limitations have important implications for the efficiency of the data transfer. Finally, we will conclude with error detection and error correction techniques.

7.2 Objectives

- To explain the design issues of Data link layer.
- To discuss about the Error detection and Error correction techniques.

7.3 Data Link Layer Design Issues

The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including:

1. Providing a well-defined service interface to the network layer.
2. Dealing with transmission errors.
3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in Fig. 7.1. Frame management forms the heart of what the data link layer does. In the following sections we will examine all the above mentioned issues in detail.

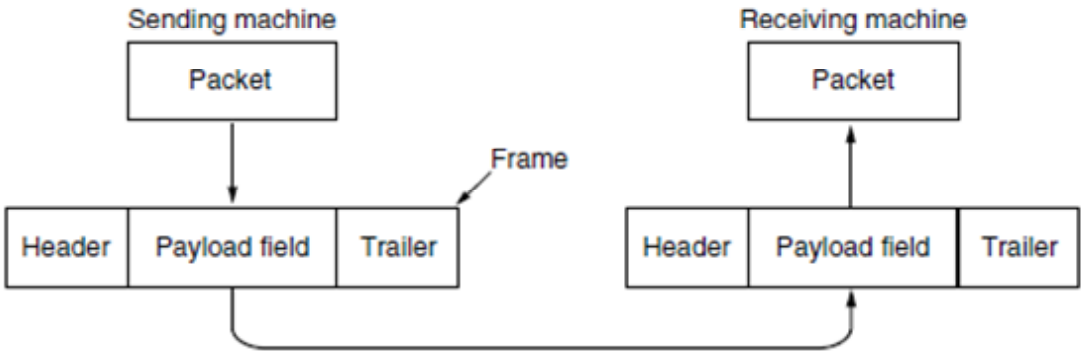


Figure 7.1 Relationship between packets and frames

7.3.1 Services Provided to the Network Layer

The function of the data link layer is to provide services to the network layer.The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine. On the source machine is an entity, call it a process, in the network

layer that hands some bits to the datalink layer for transmission to the destination. The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there, as shown in Fig. 7.2(a). The actual transmission follows the path of Fig. 7.2(b), but it is easier to think in terms of two data link layer processes communicating using a data link protocol. For this reason, we will implicitly use the model of Fig. 7.2(a) throughout this lesson.

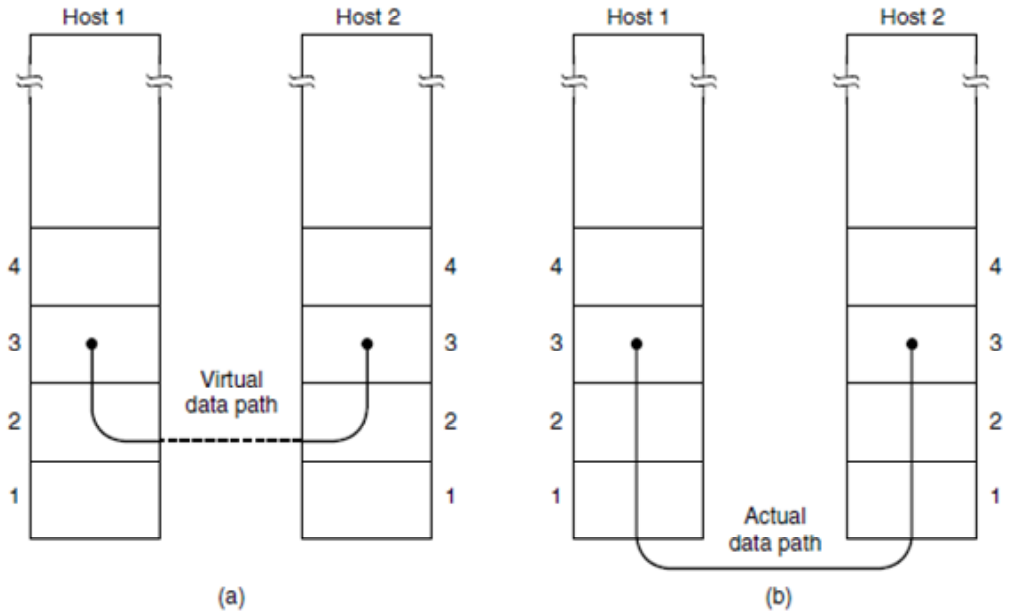


Figure 7.2(a) Virtual communication (b) Actual communication

The data link layer can be designed to offer various services. The actual services that are offered vary from protocol to protocol. Three reasonable possibilities that we will consider in turn are:

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

Unacknowledged connectionless service consists of having the source machines end independent frames to the destination machine without having the destination machine acknowledge them. Ethernet is a good example of a data link layer that provides this class of

service. No logical connection is established before hand or released afterward. If a frame is lost due to noise on the line, no attempt is made to detect the loss or recover from it in the data link layer. This class of service is appropriate when the error rate is very low, so recovery is leftto higher layers. It is also appropriate for real-time traffic, such as voice, in which late data are worse than bad data.

The next step up in terms of reliability is acknowledged connectionless service.When this service is offered, there are still no logical connections used, but each frame sent is individually acknowledged. In this way, the sender knows whether a frame has arrived correctly or been lost. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems. 802.11 (WiFi) is a good example of this classof service.

It is perhaps worth emphasizing that providing acknowledgements in the datalink layer is just an optimization, never a requirement. The network layer can always send a packet and wait for it to be acknowledged by its peer on the remote machine. If the acknowledgement is not forthcoming before the timer expires, the sender can just send the entire message again. The trouble with this strategy isthat it can be inefficient. Links usually have a strict maximum frame length imposed by the hardware, and known propagation delays. The network layer doesnot know these parameters. It might send a large packet that is broken up into,say, 10 frames, of which 2 are lost on average. It would then take a very long time for the packet to get through. Instead, if individual frames are acknowledged and retransmitted, then errors can be corrected more directly and more quickly. On reliable channels, such as fiber, the overhead of a heavyweight data link protocolmay be unnecessary, but on (inherently unreliable) wireless channels it is wellworth the cost.

The most sophisticated service the data link layer can provide to the network layer is connection-oriented service. With this service,the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received inthe right order. Connection-oriented service thus provides the network layer processes with the equivalent

of a reliable bit stream. It is appropriate over long, unreliable links such as a satellite channel or a long-distance telephone circuit. If acknowledged connectionless service were used, it is conceivable that lost acknowledgements could cause a frame to be sent and received several times, wasting bandwidth.

When connection-oriented service is used, transfers go through three distinct phases. In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not. In the second phase, one or more frames are actually transmitted. In the third and final phase, the connection is released, freeingup the variables, buffers, and other resources used to maintain the connection.

7.3.2 Framing

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. If the channel is noisy,as it is for most wireless and some wired links, the physical layer will add some redundancy to its signals to reduce the bit error rate to a tolerable level. However,the bit stream received by the data link layer is not guaranteed to be error free.Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is up to the datalink layer to detect and, if necessary, correct errors.

The usual approach is for the data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted. When a frame arrives at the destination,the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report).

Breaking up the bit stream into frames is more difficult than it at first appears.A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth. We will look at four methods:

1. Byte count.
2. Flag bytes with byte stuffing.
3. Flag bits with bit stuffing.
4. Physical layer coding violations.

The first framing method uses a field in the header to specify the number of bytes in the frame. When the data link layer at the destination sees the byte count,it knows how many bytes follow and hence where the end of the frame is. This technique is shown in Fig. 7.3(a) for four small example frames of sizes 5, 5, 8,and 8 bytes, respectively.

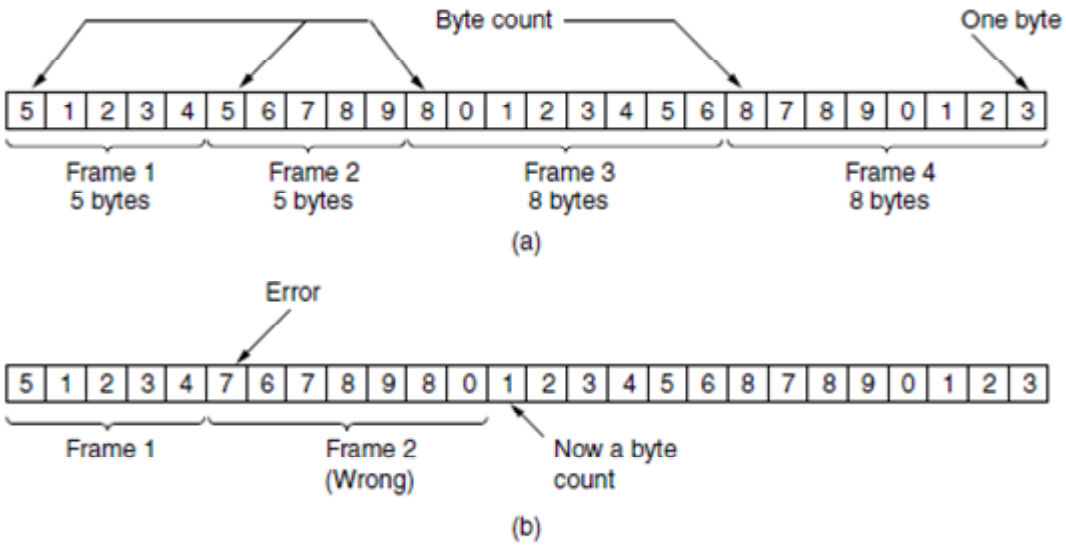


Figure 7.3 A byte stream (a) Without errors (b) With one error

The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the byte count of 5 in the second frame of Fig. 7.3(b)becomes a 7 due to a single bit flip, the destination will get out of synchronization.It will then be unable to locate the correct start of the next frame. Even if the checksum is incorrect so the destination knows that the frame is bad, it still has noway of telling where the next frame starts. Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many bytes to skip over to get to the start of the retransmission. For this reason, the byte count method is rarely used by itself.

The second framing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. Often the same byte, called a flag byte, is used as both the starting and ending delimiter.This byte is shown in Fig. 7.4(a) as FLAG. Two consecutive flag bytes indicatethe end of one frame and the start of the next. Thus, if the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame.

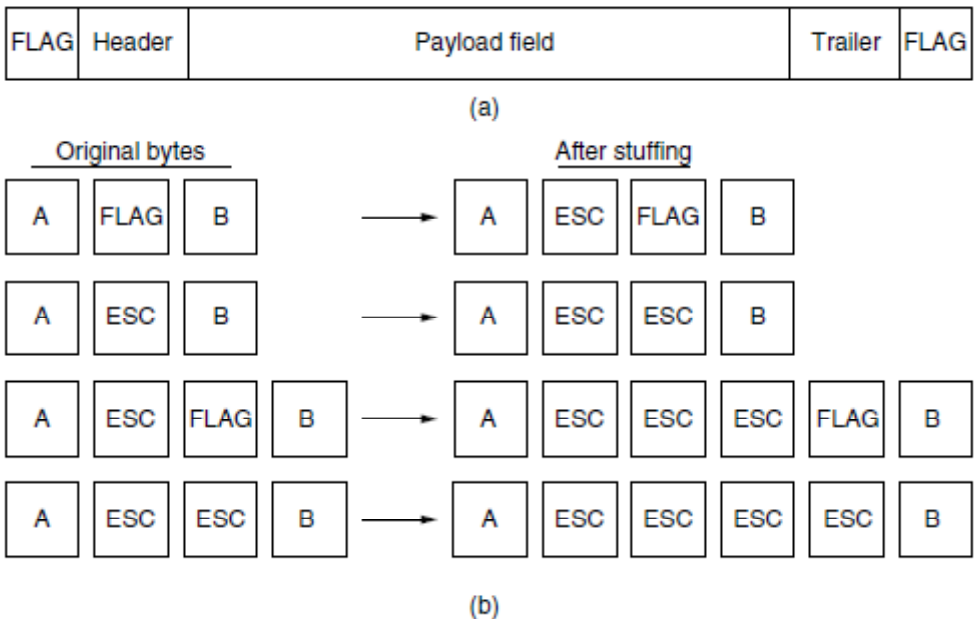


Figure 7.4(a) A frame delimited by flag bytes. (b) Four examples of byte sequences before and after byte stuffing

However, there is still a problem we have to solve. It may happen that the flag byte occurs in the data, especially when binary data such as photographs or songs are being transmitted. This situation would interfere with the framing. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each "accidental" flag byte in the data. Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it. The data link layer on the receiving end removes the escape bytes before giving the data to the network layer. This technique is called byte stuffing.

If an escape byte occurs in the middle of the data, it is stuffed with an escape byte. At the receiver, the first escape byte is removed, leaving the data byte that follows it(which might be

another escape byte or the flag byte). Some examples are shownin Fig. 7.4(b). In all cases, the byte sequence delivered after destuffing is exactly the same as the original byte sequence. We can still search for a frame boundary by looking for two flag bytes in a row, without bothering to undo escapes.

The byte-stuffing scheme depicted in Fig. 7.4 is a slight simplification of theone used in PPP (Point-to-Point Protocol), which is used to carry packets over communications links.

The third method of delimiting the bit stream gets around a disadvantage of byte stuffing, which is that it is tied to the use of 8-bit bytes. Framing can be also be done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size. It was developed for the once very popular HDLC (High level Data Link Control) protocol. Each frame begins and ends with a special bit pattern, 01111110 or 0x7E in hexadecimal. This pattern is a flag byte. Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data. It also ensures a minimum density of transitions that help the physical layer maintain synchronization. USB (Universal Serial Bus) uses bit stuffing for this reason.

When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit,it automatically destuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110. Figure 7.5 gives an example of bit stuffing.

With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern. Thus, if the receiver loses track of where it is, all it has to do is scan the input for flag sequences, since they can only occur at frame boundaries and never within the data.

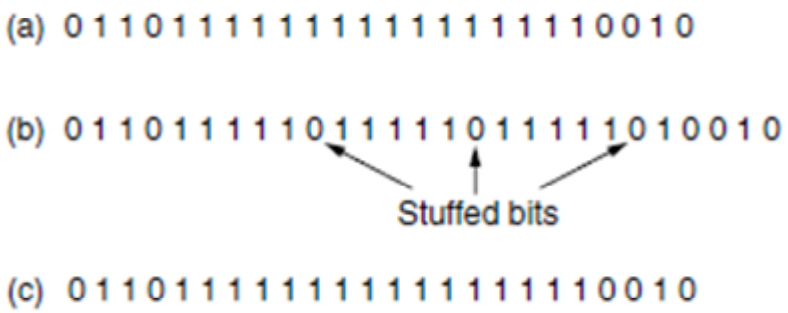


Figure 7.5 Bit stuffing (a) The original data (b) The data as they appear on the line (c) The data as they are stored in the receiver's memory after destuffing

With both bit and byte stuffing, a side effect is that the length of a frame now depends on the contents of the data it carries. For instance, if there are no flag bytes in the data, 100 bytes might be carried in a frame of roughly 100 bytes. If,however, the data consists solely of flag bytes, each flag byte will be escaped and the frame will become roughly 200 bytes long. With bit stuffing, the increase would be roughly 12.5% as 1 bit is added to every byte.

The last method of framing is to use a shortcut from the physical layer. The encoding of bits as signals often includes redundancy to help the receiver. This redundancy means that some signals will not occur in regular data. For example, in the 4B/5B line code 4 data bits are mapped to 5 signal bits to ensure sufficient bit transitions. This means that 16 out of the 32 signal possibilities are not used. We can use some reserved signals to indicate the start and end of frames. In effect, we are using "coding violations" to delimit frames.The beauty of this scheme is that, because they are reserved signals, it is easy to find the start and end of frames and there is no need to stuff the data.

Many data link protocols use a combination of these methods for safety. A common pattern used for Ethernet and 802.11 is to have a frame begin with a well-defined pattern called a preamble. This pattern might be quite long (72 bitsis typical for 802.11) to allow the receiver to prepare for an incoming packet. The preamble is then followed by a length (i.e., count) field in the header that is used to locate the end of the frame.

7.3.3 Error Control

Having solved the problem of marking the start and end of each frame, we have to make sure all frames are eventually delivered tothe network layer at the destination and in the proper order. Assume for the moment that the receiver can tell whether a frame that it receives contains correct or faulty information. For unacknowledged connectionless service it might be fine if the sender just kept outputting frames without regard to whether they were arriving properly. But for reliable, connection-oriented service it would not be fine at all.

The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line. Typically, the protocol calls for the receiver to send back special control frames bearing positive or negative acknowledgements about the incoming frames. If the sender receives a positive acknowledgement about a frame, it knows the frame has arrived safely.On the other hand, a negative acknowledgement means that something has gonewrong and the frame must be transmitted again.

An additional complication comes from the possibility that hardware troubles may cause a frame to vanish completely (e.g., in a noise burst). In this case, the receiver will not react at all, since it has no reason to react. Similarly, if the acknowledgement frame is lost, the sender will not know how to proceed. It should be clear that a protocol in which the sender transmits a frame and then waits for an acknowledgement, positive or negative, will hang forever if a frame is ever lost due to, for example, malfunctioning hardware or a faulty communication channel.

This possibility is dealt with by introducing timers into the data link layer.When the sender transmits a frame, it generally also starts a timer. The timer is set to expire after an interval long enough for the frame to reach the destination,be processed there, and have the acknowledgement propagate back to the sender.Normally, the frame will be correctly received and the acknowledgement will get back before the timer runs out, in which case the timer will be canceled.

However, if either the frame or the acknowledgement is lost, the timer will gooff, alerting the sender to a potential problem. The obvious solution is to just transmit the frame again. However, when frames may be transmitted multiple times there is a danger that the receiver will

accept the same frame two or more times and pass it to the network layer more than once. To prevent this from happening,it is generally necessary to assign sequence numbers to outgoing frames,so that the receiver can distinguish retransmissions from originals.

The whole issue of managing the timers and sequence numbers so as to ensure that each frame is ultimately passed to the network layer at the destination exactly once, no more and no less, is an important part of the duties of the data link layer(and higher layers).

7.3.4 Flow Control

Another important design issue that occurs in the data link layer (and higher layers as well) is what to do with a sender that systematically wants to transmit frames faster than the receiver can accept them. This situation can occur when the sender is running on a fast, powerful computer and the receiver is running on aslow, low-end machine. A common situation is when a smart phone requests a Web page from a far more powerful server, which then turns on the fire hose and blasts the data at the poor helpless phone until it is completely swamped. Even if the transmission is error free, the receiver may be unable to handle the frames as fast as they arrive and will lose some.

Clearly, something has to be done to prevent this situation. Two approaches are commonly used. In the first one, feedback-based flow control, the receiver sends back information to the sender giving it permission to send more data, or atleast telling the sender how the receiver is doing. In the second one, rate-based flow control, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

Feedback-based schemes are seen at both the link layer and higher layers. The latter is more common these days, in which case the link layer hardware is designed to run fast enough that it does not cause loss. For example, hardware implementations of the link layer as NICs (Network Interface Cards) are sometimes said to run at "wire speed," meaning that they can handle frames as fast as they can arrive on the link. Any overruns are then not a link problem, so they are handled by higher layers.

Various feedback-based flow control schemes are known, but most of them use the same basic principle. The protocol contains well-defined rules about when a sender may transmit the

next frame. These rules often prohibit frames from being sent until the receiver has granted permission, either implicitly or explicitly.For example, when a connection is set up the receiver might say: "You may send me n frames now, but after they have been sent, do not send any moreuntil I have told you to continue."

7.4 Error Detection and Correction

Some channels, like optical fiber in telecommunications networks, have tiny error rates so that transmission errors are a rare occurrence. But other channels,especially wireless links and aging local loops, have error rates that are orders of magnitude larger. For these links, transmission errors are the norm. They cannot be avoided at a reasonable expense or cost in terms of performance. The conclusion is that transmission errors are here to stay.

Network designers have developed two basic strategies for dealing with errors.Both add redundant information to the data that is sent. One strategy is to include enough redundant information to enable the receiver to deduce what the transmitted data must have been. The other is to include only enough redundancy to allow the receiver to deduce that an error has occurred (but not which error)and have it request a retransmission. The former strategy uses error-correcting codes and the latter uses error-detecting codes. The use of error-correcting codes is often referred to as FEC (Forward Error Correction).

Each of these techniques occupies a different ecological niche. On channels that are highly reliable, such as fiber, it is cheaper to use an error-detecting code and just retransmit the occasional block found to be faulty. However, on channels such as wireless links that make many errors, it is better to add redundancy to each block so that the receiver is able to figure out what the originally transmitted block was. FEC is used on noisy channels because retransmissions are just as likely to be in error as the first transmission.

A key consideration for these codes is the type of errors that are likely to occur.Neither error-correcting codes nor error-detecting codes can handle all possible errors since the redundant bits that offer protection are as likely to be received in error as the data bits (which can compromise their protection). It would be nice if the channel treated redundant bits differently than data bits, but it does not. They are all just bits to the channel. This means that to avoid undetected errorsthe code must be strong enough to handle the expected errors.

One model is that errors are caused by extreme values of thermal noise that overwhelm the signal briefly and occasionally, giving rise to isolated single-bit errors.Another model is that errors tend to come in bursts rather than singly. This model follows from the physical processes that generate them-such as a deep fade on a wireless channel or transient electrical interference on a wired channel.

Both models matter in practice, and they have different trade-offs. Having the errors come in bursts has both advantages and disadvantages over isolated single bit errors. On the advantage side, computer data are always sent in blocks of bits.Suppose that the block size was 1000 bits and the error rate was 0.001 per bit. If errors were independent, most blocks would contain an error. If the errors camein bursts of 100, however, only one block in 100 would be affected, on average.The disadvantage of burst errors is that when they do occur they are much harder to correct than isolated errors.

Other types of errors also exist. Sometimes, the location of an error will be known, perhaps because the physical layer received an analog signal that was far from the expected value for a 0 or 1 and declared the bit to be lost. This situation is called an erasure channel. It is easier to correct errors in erasure channels than in channels that flip bits because even if the value of the bit has been lost, at least we know which bit is in error. However, we often do not have the benefit of erasures.

Error-correcting codes are also seen in the physical layer, particularly for noisy channels, and in higher layers, particularly for real-time media and content distribution. Error-detecting codes are commonly used in link, network, and transport layers.

7.4.1 Error-Correcting Codes

We will examine four different error-correcting codes:

1. Hamming codes.
2. Binary convolutional codes.
3. Reed-Solomon codes.
4. Low-Density Parity Check codes.

All of these codes add redundancy to the information that is sent. A frame consistsof m data (i.e., message) bits and r redundant (i.e. check) bits. In a blockcode, the r check bits are computed solely as a function of the m data bits withwhich they are associated, as though the m bits were looked up in a large table to find their corresponding r check bits. In a systematic code, the m data bits aresent directly, along with the check bits, rather than being encoded themselves before they are sent. In a linear code, the r check bits are computed as a linearfunction of the m data bits. Exclusive OR (XOR) or modulo 2 addition is a popular choice. This means that encoding can be done with operations such as matrix multiplications or simple logic circuits. The codes we will look at in this section are linear, systematic block codes unless otherwise noted.

Let the total length of a block be n (i.e., n = m + r). We will describe this as an (n,m) code. An n-bit unit containing data and check bits is referred to as an n bit codeword. The code rate, or simply rate, is the fraction of the code word that carries information that is not redundant, or m/n. The rates used in practice vary widely. They might be 1/2 for a noisy channel, in which case half of the received information is redundant, or close to 1 for a high-quality channel, with only asmall number of check bits added to a large message.

To understand how errors can be handled, it is necessary to first look closely at what an error really is. Given any two codewords that may be transmitted or received-say, 10001001 and 10110001-it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, just XOR the two codewords and count the number of 1 bits in the result.For example:

10001001

10110001

00111000

The number of bit positions in which two codewords differ is called the Hamming distance (Hamming, 1950). Its significance is that if two codewords are a Hamming distance d apart, it will require d single-bit errors to convert one into the other.

Given the algorithm for computing the check bits, it is possible to construct a complete list of the legal codewords, and from this list to find the two codewords with the smallest Hamming distance. This distance is the Hamming distance of the complete code.

In most data transmission applications, all 2m possible data messages arelegal, but due to the way the check bits are computed, not all of the 2n possible codewords are used. In fact, when there are r check bits, only the small fraction of 2m/2n or 1/2r of the possible messages will be legal codewords. It is the sparseness with which the message is embedded in the space of codewords that allows the receiver to detect and correct errors.

The error-detecting and error-correcting properties of a block code depend on its Hamming distance. To reliably detect d errors, you need a distance d + 1 code because with such a code there is no way that d single-bit errors can change a valid codeword into another valid codeword. When the receiver sees an illegal codeword, it can tell that a transmission error has occurred. Similarly, to correct derrors, you need a distance 2d + 1 code because that way the legal codewords are so far apart that even with d changes the original codeword is still closer than any other codeword. This means the original codeword can be uniquely determined based on the assumption that a larger number of errors are less likely.

As a simple example of an error-correcting code, consider a code with only four valid codewords:

0000000000, 0000011111, 1111100000, and 1111111111

This code has a distance of 5, which means that it can correct double errors or detect quadruple errors. If the codeword 0000000111 arrives and we expect only single- or double-bit errors, the receiver will know that the original must have been 0000011111. If, however, a triple error changes 0000000000 into 0000000111, the error will not be corrected properly. Alternatively, if we expect all of these errors, we can detect them. None of the received codewords are legal codewords so an error must have occurred. It should be apparent that in this example we cannot both correct double errors and detect quadruple errors because this would require us to interpret a received codeword in two different ways.

In Hamming codes the bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and soon. The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits. The rest (3,5, 6, 7, 9, etc.) are filled up with the m data bits. This pattern is shown for an(11,7) Hamming code with 7 data bits and 4 check bits in Fig. 7.6. Each check bit forces the modulo 2 sum, or parity, of some collection of bits, including itself, to be even (or odd). A bit may be included in several check bit computations. To see which check bits the data bit in position k contributes to, rewrite k as a sum of powers of 2. For example, 11 = 1 + 2 + 8 and 29 = 1 + 4 + 8 + 16. A bit is checked by just those check bits occurring in its expansion (e.g., bit 11 is checked by bits 1, 2, and 8). In the example, the check bits are computed for even parity sums for a message that is the ASCII letter "A."

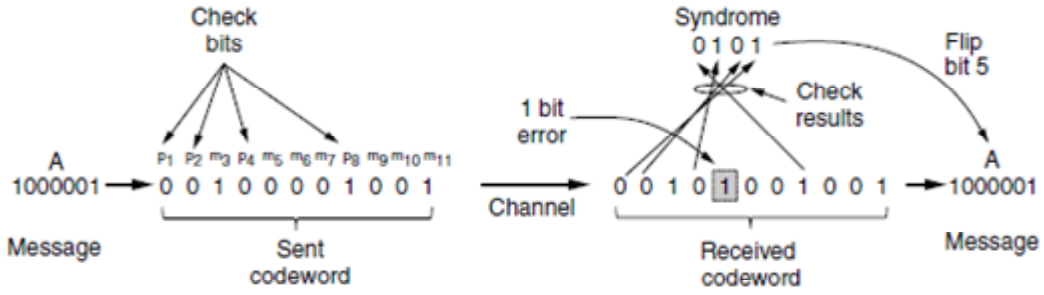


Figure 7.6 Example of an (11, 7) Hamming code correcting a single-bit error

This construction gives a code with a Hamming distance of 3, which means that it can correct single errors (or detect double errors). The reason for the very careful numbering of message and check bits becomes apparent in the decoding process. When a codeword arrives, the receiver redoes the check bit computations including the values of the received check bits. We call these the check results.

If the check bits are correct then, for even parity sums, each check result should be zero. In this case the codeword is accepted as valid.If the check results are not all zero, however, an error has been detected. The set of check results forms the error syndrome that is used to pinpoint and correct the error. In Fig. 7.6, a single-bit error occurred on the channel so the check results are 0, 1, 0, and 1 for k = 8, 4, 2, and 1, respectively. This gives a syndrome of 0101 or 4 + 1=5. By the design of the scheme, this means that the fifth bit is inerror. Flipping the incorrect bit (which might be a check bit or a data bit) and discarding the check bits gives the correct message of an ASCII "A."

Hamming distances are valuable for understanding block codes, and Hamming codes are used in error-correcting memory. However, most networks use stronger codes. The second code we will look at is a convolutional code. In a convolutional code, an encoder processes a sequence of input bits and generates a sequence of output bits. There is no natural message size or encoding boundary as in a blockcode. The output depends on the current and previous input bits. That is, the encoder has memory. The number of previous bits on which the output depends is called the constraint length of the code. Convolutional codes are specified in terms of their rate and constraint length.

Convolutional codes are widely used in deployed networks, for example, as part of the GSM mobile phone system, in satellite communications, and in 802.11.As an example, a popular convolutional code is shown in Fig. 7.7. This code isknown as the NASA convolutional code of r = 1/2 and k = 7, since it was first used for the Voyager space missions starting in 1977. Since then it has beenliberally reused, for example, as part of 802.11.

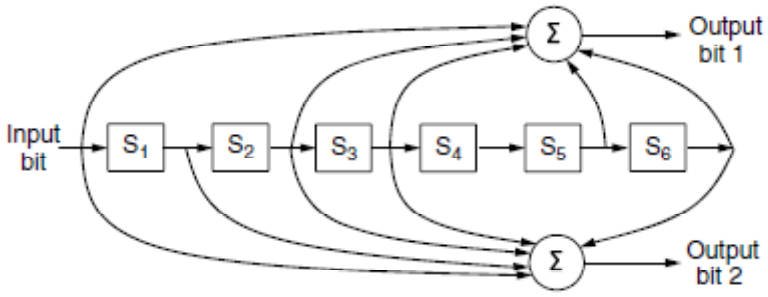


Figure 7.7 The NASA binary convolutional code used in 802.11

In Fig. 7.7, each input bit on the left-hand side produces two output bits on the right-hand side that are XOR sums of the input and internal state. Since it deals with bits and performs linear operations, this is a binary, linear convolutional code. Since 1 input bit produces 2 output bits, the code rate is 1/2. It is not systematic since none of the output bits is simply the input bit.

The internal state is kept in six memory registers. Each time another bit is input the values in the registers are shifted to the right. For example, if 111 is input and the initial state is all zeros, the internal state, written left to right, will become 100000, 110000, and 111000 after the first, second, and third bits have been input. The output bits will be 11, followed by 10, and then

01. It takes seven shifts to flush an input completely so that it does not affect the output. The constraint length of this code is thus k = 7.

A convolutional code is decoded by finding the sequence of input bits that is most likely to have produced the observed sequence of output bits (which includes any errors). For small values of k, this is done with a widely used algorithm developed by Viterbi (Forney, 1973). The algorithm walks the observed sequence, keeping for each step and for each possible internal state the input sequence that would have produced the observed sequence with the fewest errors. The input sequence requiring the fewest errors at the end is the most likely message.

Convolutional codes have been popular in practice because it is easy to factor the uncertainty of a bit being a 0 or a 1 into the decoding. For example, suppose –1V is the logical 0 level and +1V is the logical 1 level, we might receive 0.9V and –0.1V for 2 bits. Instead of mapping these signals to 1 and 0 right away, we would like to treat 0.9V as "very likely a 1" and –0.1V as "maybe a 0" and correct the sequence as a whole. Extensions of the Viterbi algorithm can work with these uncertainties to provide stronger error correction. This approach of working with the uncertainty of a bit is called soft-decision decoding. Conversely, deciding whether each bit is a 0 or a 1 before subsequent error correction is called hard-decision decoding.

The third kind of error-correcting code is the Reed-Solomon code. Like Hamming codes, Reed-Solomon codes are linear block codes, and they are often systematic too. Unlike Hamming codes, which operate on individual bits, Reed-Solomon codes operate on m bit symbols. Naturally, the mathematics are more involved, so we will describe their operation by analogy.

Reed-Solomon codes are based on the fact that every n degree polynomial is uniquely determined by n + 1 points. For example, a line having the form ax + b is determined by two points. Extra points on the same line are redundant, which is helpful for error correction. Imagine that we have two data points that represent a line and we send those two data points plus two check points chosen to lie on the same line. If one of the points is received in error, we can still recover the data points by fitting a line to the received points. Three of the points will lie on the line, and one point, the one in error, will not. By finding the line we have corrected the error.

Reed-Solomon codes are actually defined as polynomials that operate over finite fields, but they work in a similar manner. For m bit symbols, the codewords are 2m–1symbols long. A

popular choice is to make m = 8 so that symbols are bytes. A codeword is then 255 bytes long. The (255, 233) code is widely used; it adds 32 redundant symbols to 233 data symbols. Decoding with error correctionis done with an algorithm developed by Berlekamp and Massey that can efficiently perform the fitting task for moderate-length codes (Massey, 1969).

Reed-Solomon codes are widely used in practice because of their strong error-correction properties, particularly for burst errors. They are used for DSL, data over cable, satellite communications, and perhaps most ubiquitously on CDs, DVDs, and Blu-ray discs. Because they are based on m bit symbols, a single-bit error and an m-bit burst error are both treated simply as one symbol error. When 2t redundant symbols are added, a Reed-Solomon code is able to correct up to t errors in any of the transmitted symbols. This means, for example, that the (255, 233) code, which has 32 redundant symbols, can correct up to 16 symbol errors. Since the symbols may be consecutive and they are each 8 bits, an error burst of up to 128 bits can be corrected. The situation is even better if the error model is one of erasures (e.g., a scratch on a CD that obliterates some symbols). In this case, up to 2t errors can be corrected.

Reed-Solomon codes are often used in combination with other codes such as a convolutional code. The thinking is as follows. Convolutional codes are effective at handling isolated bit errors, but they will fail, likely with a burst of errors, if there are too many errors in the received bit stream. By adding a Reed-Solomon code within the convolutional code, the Reed-Solomon decoding can mop up the error bursts, a task at which it is very good. The overall code then provides good protection against both single and burst errors.

The final error-correcting code is the LDPC (Low-Density Parity Check) code. LDPC codes are linear block codes that were invented by Robert Gallagher in the year 1962. In an LDPC code, each output bit is formed from only a fraction of the input bits. This leads to a matrix representation of the code that has a low density of 1s, hence the name for the code. The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword. This corrects errors.

LDPC codes are practical for large block sizes and have excellent error-correction abilities that outperform many other codes in practice. For this reason they are rapidly being included in new protocols. They are part of the standard for digital video broadcasting, 10 Gbps Ethernet, power-line networks, and the latest version of 802.11.

7.4.2 Error-Detecting Codes

Error-correcting codes are widely used on wireless links, which are notoriously noisy and error prone when compared to optical fibers. Without error-correcting codes, it would be hard to get anything through. However, over fiber orhigh-quality copper, the error rate is much lower, so error detection and retransmission is usually more efficient there for dealing with the occasional error. We will examine three different error-detecting codes. They are all linear, systematic block codes: 1. Parity. 2. Checksums. 3. Cyclic Redundancy Checks (CRCs).

To see how they can be more efficient than error-correcting codes, consider the first error-detecting code, in which a single parity bit is appended to the data. The parity bit is chosen so that the number of 1 bits in the codeword is even (or odd). Doing this is equivalent to computing the (even) parity bit as the modulo 2 sum or XOR of the data bits. For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100. With odd parity 1011010 becomes 10110101. A code with a single parity bit has a distance of 2, since any single-bit error produces a codeword with the wrong parity. This means that it can detect single-bit errors.

One difficulty with this scheme is that a single parity bit can only reliably detect a single-bit error in the block. If the block is badly garbled by a long burst error, the probability that the error will be detected is only 0.5, which is hardly acceptable. The odds can be improved considerably if each block to be sent is regarded as a rectangular matrix n bits wide and k bits high. Now, if we compute and send one parity bit for each row, up to k bit errors will be reliably detected as long as there is at most one error per row.

However, there is something else we can do that provides better protection against burst errors: we can compute the parity bits over the data in a different order than the order in which the data bits are transmitted. Doing so is called interleaving. In this case, we will compute a parity bit for each of the n columns and send all the data bits as k rows, sending the rows from top to bottom and the bits in each row from left to right in the usual manner. At the last row, we send the n parity bits. This transmission order is shown in Fig. 7.8 for n = 7 and k = 7.

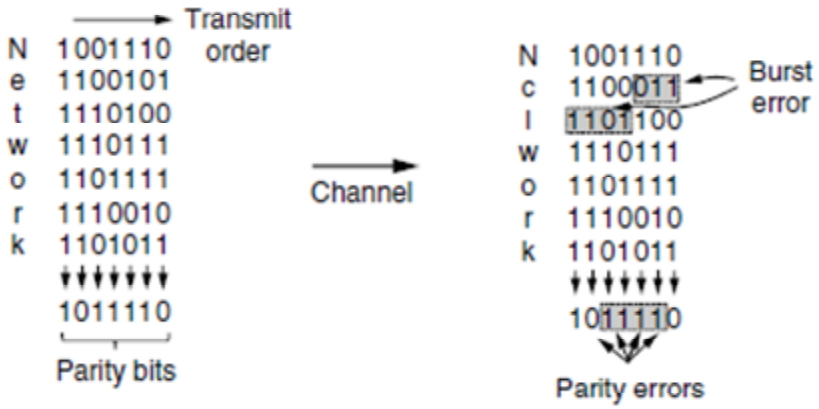


Figure 7.8 Interleaving of parity bits to detect a burst error

Interleaving is a general technique to convert a code that detects (or corrects)isolated errors into a code that detects (or corrects) burst errors. In Fig. 7.8, when a burst error of length n = 7 occurs, the bits that are in error are spread across different columns. (A burst error does not imply that all the bits are wrong; it just implies that at least the first and last are wrong. In Fig. 7.8, 4 bits were flipped over a range of 7 bits.) At most 1 bit in each of the n columns will be affected, so the parity bits on those columns will detect the error. This method uses n parity bits on blocks of n data bits to detect a single burst error of length n or less.

A burst of length n + 1 will pass undetected, however, if the first bit is inverted, the last bit is inverted, and all the other bits are correct. If the block is badly garbled by a long burst or by multiple shorter bursts, the probability that any of the n columns will have the correct parity by accident is 0.5, so the probability of a bad block being accepted when it should not be is 2–n.

The second kind of error-detecting code, the checksum, is closely related to groups of parity bits. The word "checksum" is often used to mean a group of check bits associated with a message, regardless of how are calculated. A group of parity bits is one example of a checksum. However, there are other, stronger checksums based on a running sum of the data bits of the message. The checksum is usually placed at the end of the message, as the complement of the sum function.This way, errors may be detected by summing the entire received codeword,both data bits and checksum. If the result comes out to be zero, no error has been detected.

One example of a checksum is the 16-bit Internet checksum used on all Internetpackets as part of the IP protocol (Braden et al., 1988). This checksum is a sum of the message bits divided into 16-bit words. Because this method operates on words rather than on bits, as in parity, errors that leave the parity unchanged can still alter the sum and be detected. For example, if the lowest order bit in two different words is flipped from a 0 to a 1, a parity check across these bits would fail to detect an error. However, two 1s will be added to the 16-bit checksum to produce a different result. The error can then be detected.

The Internet checksum in particular is efficient and simple but provides weak protection in some cases precisely because it is a simple sum. It does not detect the deletion or addition of zero data, nor swapping parts of the message, and it provides weak protection against message splices in which parts of two packets are put together. These errors may seem very unlikely to occur by random processes,but they are just the sort of errors that can occur with buggy hardware.

A better choice is Fletcher's checksum (Fletcher, 1982). It includes a positional component, adding the product of the data and its position to the running sum. This provides stronger detection of changes in the position of data.

Although the two preceding schemes may sometimes be adequate at higher layers, in practice, a third and stronger kind of error-detecting code is in widespread use at the link layer: the CRC (Cyclic Redundancy Check), also known as a polynomial code. Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. A k-bit frame is regarded as the coefficient list for a polynomial with k terms, ranging from x k - 1to x 0. Such a polynomial is said to be of degree k – 1. The high-order (leftmost)bit is the coefficient of x k – 1, the next bit is the coefficient of x k – 2, and so on.For example, 110001 has 6 bits and thus represents a six-term polynomial withcoefficients 1, 1, 0, 0, 0, and 1: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$.

Polynomial arithmetic is done modulo 2, according to the rules of algebraic field theory. It does not have carries for addition or borrows for subtraction. Both addition and subtraction are identical to exclusive OR. For example:

10011011 00110011 11110000 01010101

+ 11001010 + 11001101 ? 10100110 ? 10101111

01010001 11111110 01010110 11111010

Long division is carried out in exactly the same way as it is in binary except that the subtraction is again done modulo 2. A divisor is said "to go into" a dividend if the dividend has as many bits as the divisor.

When the polynomial code method is employed, the sender and receiver must agree upon a generator polynomial, G(x), in advance. Both the high- and low order bits of the generator must be 1. To compute the CRC for some frame with m bits corresponding to the polynomial M(x), the frame must be longer than the generator polynomial. The idea is to append a CRC to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisibleby G(x). When the receiver gets the checksummed frame, it tries dividing itby G(x). If there is a remainder, there has been a transmission error.

The algorithm for computing the CRC is as follows:

1. Let r be the degree of G(x). Append r zero bits to the low-order end of the frame so it now contains m + r bits and corresponds to the polynomial x rM(x).
2. Divide the bit string corresponding to G(x) into the bit string corresponding to x rM(x), using modulo 2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to x rM(x) using modulo 2 subtraction. The resultis the checksummed frame to be transmitted. Call its polynomialT(x).

Among other desirable properties, it has the property that it detects all bursts of length 32 or less and all bursts affecting an odd number of bits. It has been used widely since the 1980s. However, this does not mean it is the best choice. Using an exhaustive computational search, Castagnoli et al. (1993) and Koopman (2002)found the best CRCs. These CRCs have a Hamming distance of 6 for typical message sizes, while the IEEE standard CRC-32 has a Hamming distance of only4.

Although the calculation required to compute the CRC may seem complicated,it is easy to compute and verify CRCs in hardware with simple shift registe rcircuits (Peterson and Brown, 1961). In practice, this hardware is nearly always used. Dozens of networking standards include various CRCs, including virtually all LANs (e.g., Ethernet, 802.11) and point-to-point links (e.g., packets over SONET).

7.5 Summary

The task of the data link layer is to convert the raw bit stream offered by the physical layer into a stream of frames for use by the network layer. The link layer can present this stream with varying levels of reliability, ranging from connectionless,unacknowledged service to reliable, connection-oriented service.

Various framing methods are used, including byte count, byte stuffing, and bit stuffing. Data link protocols can provide error control to detect or correct damaged frames and to retransmit lost frames. To prevent a fast sender from overrunning a slow receiver, the data link protocol can also provide flow control. The sliding window mechanism is widely used to integrate error control and flow controlin a simple way. When the window size is 1 packet, the protocol is stop-and-wait.

Codes for error correction and detection add redundant information to messages by using a variety of mathematical techniques. Convolutional codes and Reed-Solomon codes are widely deployed for error correction, with low-density parity check codes increasing in popularity. The codes for error detection that areused in practice include cyclic redundancy checks and checksums. All these codes can be applied at the link layer, as well as at the physical layer and higher layers.

7.6 Model Questions

1. Write notes on Framing in Data link layer.
2. Explain the design issues of Data link layer.
3. Discuss on Error detection and Error correction techniques.
4. Explain Error correcting codes with example.
5. Discuss about the services provided by Data link layer to the Network layer.

LESSON-8
DATA LINK LAYER PROTOCOLS

Structure

8.1 Introduction

Protocols in the data link layer are designed so that this layer can perform its basic functions: framing, error control and flow control. Framing is the process of dividing bit - streams from physical layer into data frames whose size ranges from a few hundred to a few thousand bytes. Error control mechanisms deals with transmission errors and retransmission of corrupted and lost frames. Flow control regulates speed of delivery and so that a fast sender does not drown a slow receiver.To ensure that frames are delivered free of errors to the destination station a number of requirements are placed on a data link protocol. Let us discuss about the types of protocols in data link layer.

8.2 Objectives

- To explain about the Elementary data link protocols and it's types.
- To discuss about the Sliding window protocols and it's types.

8.3 Elementary Data Link Protocols

To introduce the subject of protocols, we will begin by looking at three protocols of increasing complexity. Before we look at the protocols, it is useful to make explicit some of the assumptions underlying the model of communication.

To start with, we assume that the physical layer, data link layer, and network layer are independent processes that communicate by passing messages back and forth. A common implementation is shown in Fig. 8.1. The physical layer process and some of the data link layer process run on dedicate hardware called a NIC (Network Interface Card). The rest of the link layer process and the network layer process run on the main CPU as part of the operating system, with the software for the link layer process often taking the form of a device driver. However, other implementations are also possible (e.g., three processes offloaded to dedicated hardware called a network accelerator, or three processes running on the main CPU on a software-defined ratio). Actually, the preferred implementation changes from decade to decade with technology trade-offs. In any event, treating the three layers as separate processes makes the discussion conceptually cleaner and also serves to emphasize the independence of the layers.

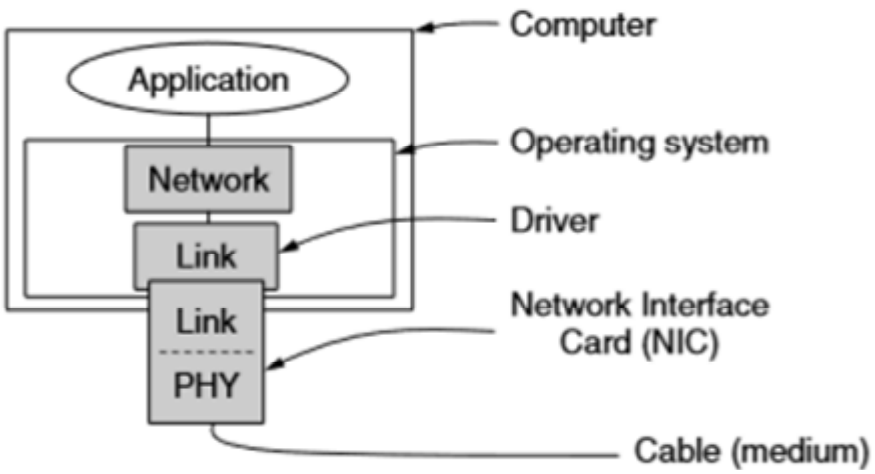


Figure 8.1 Implementation of the physical, data link, and network layers

Another key assumption is that machine A wants to send a long stream of data to machine B, using a reliable, connection-oriented service. Later, we will consider the case where B also wants to send data to A simultaneously. A is assumed to have an infinite supply of data ready to send and never has to wait for data to be produced. Instead, when A's data link layer asks for data, the network layer is always able to comply immediately.

As far as the data link layer is concerned, the packet passed across the interface to it from the network layer is pure data, whose every bit is to be delivered to the destination's

network layer. The fact that the destination's network layer may interpret part of the packet as a header is of no concern to the data link layer.

When the data link layer accepts a packet, it encapsulates the packet in a frame by adding a data link header and trailer to it. Thus, a frame consists of an embedded packet, some control information (in the header), and a checksum (in the trailer). The frame is then transmitted to the data link layer on the other machine. We will assume that there exist suitable library procedures to physical layer to send a frame and from physical layer to receive a frame.

Initially, the receiver has nothing to do. It just sits around waiting for something to happen. In the example protocols throughout this chapter we will indicate that the data link layer is waiting for something to happen by the procedure callwait for event(&event). This procedure only returns when something has happened (e.g., a frame has arrived). Upon return, the variable event tells what happened. The set of possible events differs for the various protocols to be described and will be defined separately for each protocol. Note that in a more realistic situation, the data link layer will not sit in a tight loop waiting for an event, as we have suggested, but will receive an interrupt, which will cause it to stop whatever it was doing and go handle the incoming frame. Nevertheless, for simplicity we will ignore all the details of parallel activity within the data link layer and assume that it is dedicated full time to handling just our one channel.

When a frame arrives at the receiver, the checksum is recomputed. If the checksum in the frame is incorrect (i.e., there was a transmission error), the data link layer is so informed (event = cksum_err). If the inbound frame arrived undamaged, the data link layer is also informed (event = frame arrival) so that it can acquire the frame for inspection using from physical layer. As soon as the receiving data link layer has acquired an undamaged frame, it checks the control information in the header, and, if everything is all right, passes the packet portion to the network layer. Under no circumstances is a frame header ever given to a network layer.

There is a good reason why the network layer must never be given any part of the frame header: to keep the network and data link protocols completely separate. As long as the network layer knows nothing at all about the data link protocol or the frame format, these things can be changed without requiring changes to the network layer's software. This happens whenever a

new NIC is installed in a computer. Providing a rigid interface between the network and data link layers greatly simplifies the design task because communication protocols in different layers can evolve independently.

A frame is composed of four fields: kind, seq, ack, and info, the first three of which contain control information and the last of which may contain actual data to be transferred. These control fields are collectively called the frame header.

The kind field tells whether there are any data in the frame, because some of the protocols distinguish frames containing only control information from those containing data as well. The seq and ack fields are used for sequence numbers and acknowledgements, respectively. The info field of a data frame contains a single packet; the info field of a control frame is not used. A more realistic implementation would use a variablelength info field, omitting it altogether for control frames.

Again, it is important to understand the relationship between a packet and a frame. The network layer builds a packet by taking a message from the transport layer and adding the network layer header to it. This packet is passed to the data link layer for inclusion in the info field of an outgoing frame. When the frame arrives at the destination, the data link layer extracts the packet from the frame and passes the packet to the network layer. In this manner, the network layer can act as though machines can exchange packets directly.

In most of the protocols, we assume that the channel is unreliable and loses entire frames upon occasion. To be able to recover from such calamities, the sending data link layer must start an internal timer or clock whenever it sends a frame. If no reply has been received within a certain predetermined time interval, the clock times out and the data link layer receives an interrupt signal.

In our protocols this is handled by allowing the procedure wait_for_event to return event = timeout. The procedures start_timer and stop_timer turn the timer on and off, respectively. Timeout events are possible only when the timer is running and before stop_timer is called. It is explicitly permitted to call start_timer while the timer is running; such a call simply resets the clock to cause the next timeout after a full timer interval has elapsed (unless it is reset or turned off).

The procedures start_ack_timer and stop_ack_timer control an auxiliary timer used to generate acknowledgements under certain conditions. The procedures enable network layer and disable network layer are used in the more sophisticated protocols, where we no longer assume that the network layer always has packets to send. When the data link layer enables the network layer, the network layer is then permitted to interrupt when it has a packet to be sent. We indicate this with event = network_layer_ready. When the network layer is disabled, it may not cause such events. By being careful about when it enables and disables its network layer, the data link layer can prevent the network layer from swamping it with packets for which it has no buffer space.

Frame sequence numbers are always in the range 0 to MAX_SEQ (inclusive), where MAX_SEQ is different for the different protocols. It is frequently necessaryto advance a sequence number by 1 circularly (i.e., MAX_SEQ is followed by 0). The macro inc performs this incrementing. It has been defined as a macro because it is used in-line within the critical path.

8.3.1 A Utopian Simplex Protocol

As an initial example we will consider a protocol that is as simple as it can be because it does not worry about the possibility of anything going wrong. Data are transmitted in one direction only. Both the transmitting and receiving network layers are always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This thoroughly unrealistic protocol, which we will nickname "Utopia," is simply to show the basic structure on which we will build. It's implementation is shown in Fig. 8.2.

```
/* Protocol 1 (Utopia) provides for data transmission in one direction only, from
   sender to receiver. The communication channel is assumed to be error free
   and the receiver is assumed to be able to process all the input infinitely quickly.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                          /* buffer for an outbound frame */
  packet buffer;                    /* buffer for an outbound packet */

  while (true) {
      from_network_layer(&buffer);  /* go get something to send */
      s.info = buffer;              /* copy it into s for transmission */
      to_physical_layer(&s);        /* send it on its way */
  }                                 /* Tomorrow, and tomorrow, and tomorrow,
                                       Creeps in this petty pace from day to day
                                       To the last syllable of recorded time.
                                           – Macbeth, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event;                 /* filled in by wait, but not used here */

  while (true) {
      wait_for_event(&event);       /* only possibility is frame_arrival */
      from_physical_layer(&r);      /* go get the inbound frame */
      to_network_layer(&r.info);    /* pass the data to the network layer */
  }
}
```

Figure 8.2 A utopian simplex protocol

The protocol consists of two distinct procedures, a sender and a receiver. The sender runs in the data link layer of the source machine, and the receiver runs in the data link layer of the destination machine. No sequence numbers or acknowledgements are used here, so MAX_SEQ is not needed. The only event type possible is frame_arrival (i.e., the arrival of an undamaged frame).

The sender is in an infinite while loop just pumping data out onto the line as fast as it can. The body of the loop consists of three actions: go fetch a packet from the (always obliging)

network layer, construct an outbound frame using the variable s, and send the frame on its way. Only the info field of the frame is used by this protocol, because the other fields have to do with error and flow control and there are no errors or flow control restrictions here.

The receiver is equally simple. Initially, it waits for something to happen, the only possibility being the arrival of an undamaged frame. Eventually, the frame arrives and the procedure wait_for_event returns, with event set to frame_arrival(which is ignored anyway). The call to from physical layer removes the newly arrived frame from the hardware buffer and puts it in the variable r, where the receiver code can get at it. Finally, the data portion is passed on to the network layer, and the data link layer settles back to wait for the next frame, effectively suspending itself until the frame arrives.

The utopia protocol is unrealistic because it does not handle either flow control or error correction. Its processing is close to that of an unacknowledged connectionless service that relies on higher layers to solve these problems, though even an unacknowledged connectionless service would do some error detection.

8.3.2 A Simplex Stop-and-Wait Protocol for an Error-Free Channel

Now we will tackle the problem of preventing the sender from flooding the receiver with frames faster than the latter is able to process them. This situation can easily happen in practice so being able to prevent it is of great importance.The communication channel is still assumed to be error free, however, and the data traffic is still simplex.

One solution is to build the receiver to be powerful enough to process a continuous stream of back-to-back frames (or, equivalently, define the link layer to be slow enough that the receiver can keep up). It must have sufficient buffering and processing abilities to run at the line rate and must be able to pass the frames that are received to the network layer quickly enough. However, this is a worst-case solution. It requires dedicated hardware and can be wasteful of resources if the utilization of the link is mostly low. Moreover, it just shifts the problem of dealing with a sender that is too fast elsewhere; in this case to the network layer.

A more general solution to this problem is to have the receiver provide feedback to the sender. After having passed a packet to its network layer, the receiver sends a little dummy

frame back to the sender which, in effect, gives the sender permission to transmit the next frame. After having sent a frame, the sender is required by the protocol to bide its time until the little dummy (i.e., acknowledgement) frame arrives. This delay is a simple example of a flow control protocol. Protocols in which the sender sends one frame and then waits for an acknowledgement before proceeding are called stop-and-wait. Figure 8.3 gives an example of a simplex stop-and-wait protocol.

```
/* Protocol 2 (Stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
 frame s;                          /* buffer for an outbound frame */
 packet buffer;                    /* buffer for an outbound packet */
 event_type event;                 /* frame_arrival is the only possibility */

 while (true) {
     from_network_layer(&buffer);  /* go get something to send */
     s.info = buffer;              /* copy it into s for transmission */
     to_physical_layer(&s);        /* bye-bye little frame */
     wait_for_event(&event);       /* do not proceed until given the go ahead */
 }
}

void receiver2(void)
{
 frame r, s;                       /* buffers for frames */
 event_type event;                 /* frame_arrival is the only possibility */
 while (true) {
     wait_for_event(&event);       /* only possibility is frame_arrival */
     from_physical_layer(&r);      /* go get the inbound frame */
     to_network_layer(&r.info);    /* pass the data to the network layer */
     to_physical_layer(&s);        /* send a dummy frame to awaken sender */
 }
}
```

Figure 8.3 A simplex stop-and-wait protocol

Although data traffic in this example is simplex, going only from the sender to the receiver, frames do travel in both directions. Consequently, the communication channel between the two data link layers needs to be capable of bidirectional information transfer. However, this protocol entails a strict alternation of flow: first the sender sends a frame, then the receiver sends a

frame, then the sender sends another frame, then the receiver sends another one, and so on. A half duplex physical channel would suffice here.

As in protocol 1, the sender starts out by fetching a packet from the network layer, using it to construct a frame, and sending it on its way. But now, unlike in protocol 1, the sender must wait until an acknowledgement frame arrives before looping back and fetching the next packet from the network layer. The sending data link layer need not even inspect the incoming frame as there is only one possibility. The incoming frame is always an acknowledgement.

The only difference between receiver1 and receiver2 is that after delivering a packet to the network layer, receiver2 sends an acknowledgement frame back to the sender before entering the wait loop again. Because only the arrival of the frame back at the sender is important, not its contents, the receiver need not put any particular information in it.

8.3.3 A Simplex Stop-and-Wait Protocol for a Noisy Channel

Now let us consider the normal situation of a communication channel that makes errors. Frames may be either damaged or lost completely. However, we assume that if a frame is damaged in transit, the receiver hardware will detect this when it computes the checksum. If the frame is damaged in such a way that the checksum is nevertheless correct-an unlikely occurrence-this protocol (and all other protocols) can fail (i.e., deliver an incorrect packet to the network layer).

At first glance it might seem that a variation of protocol 2 would work: adding a timer. The sender could send a frame, but the receiver would only send an acknowledgement frame if the data were correctly received. If a damaged frame arrived at the receiver, it would be discarded. After a while the sender would time out and send the frame again. This process would be repeated until the frame finally arrived intact.

To see what might go wrong, remember that the goal of the data link layer is to provide error-free, transparent communication between network layer processes. The network layer on machine A gives a series of packets to its data link layer, which must ensure that an identical series of packets is delivered to the network layer on machine B by its data link layer. In particular,

the network layer on B has no way of knowing that a packet has been lost or duplicated, so the data link layer must guarantee that no combination of transmission errors, however unlikely, can cause a duplicate packet to be delivered to a network layer.

Clearly, what is needed is some way for the receiver to be able to distinguish a frame that it is seeing for the first time from a retransmission. The obvious way to achieve this is to have the sender put a sequence number in the header of each frame it sends. Then the receiver can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded.

Since the protocol must be correct and the sequence number field in the header is likely to be small to use the link efficiently, the question arises: what is the minimum number of bits needed for the sequence number? The header might provide 1 bit, a few bits, 1 byte, or multiple bytes for a sequence number depending on the protocol. The important point is that it must carry sequence numbers that are large enough for the protocol to work correctly, or it is not much of a protocol.

The only ambiguity in this protocol is between a frame, m, and its direct successor, m + 1. If frame m is lost or damaged, the receiver will not acknowledge it, so the sender will keep trying to send it. Once it has been correctly received, the receiver will send an acknowledgement to the sender. It is here that the potential trouble crops up. Depending upon whether the acknowledgement frame gets back to the sender correctly or not, the sender may try to send m or m + 1.

At the sender, the event that triggers the transmission of frame m + 1 is the arrival of an acknowledgement for frame m. But this situation implies that m – 1 has been correctly received, and furthermore that its acknowledgement has also been correctly received by the sender. Otherwise, the sender would not have begun with m, let alone have been considering m + 1. As a consequence, the only ambiguity is between a frame and its immediate predecessor or successor, not between the predecessor and successor themselves.

A 1-bit sequence number (0 or 1) is therefore sufficient. At each instant of time, the receiver expects a particular sequence number next. When a frame containing the correct sequence number arrives, it is accepted and passed to the network layer, then acknowledged.

Then the expected sequence number is incremented modulo 2 (i.e., 0 becomes 1 and 1 becomes 0). Any arriving frame containing the wrong sequence number is rejected as a duplicate. However, the last valid acknowledgement is repeated so that the sender can eventually discover that the frame has been received.

An example of this kind of protocol is shown in Fig. 8.4. Protocols in which the sender waits for a positive acknowledgement before advancing to the next data item are often called ARQ (Automatic Repeat reQuest) or PAR (Positive Acknowledgement with Retransmission). Like protocol 2, this one also transmits data only in one direction.

```
/* Protocol 3 (PAR) allows unidirectional data flow over an unreliable channel. */

#define MAX_SEQ 1                                   /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                        /* seq number of next outgoing frame */
  frame s;                                          /* scratch variable */
  packet buffer;                                    /* buffer for an outbound packet */
  event_type event;

  next_frame_to_send = 0;                           /* initialize outbound sequence numbers */
  from_network_layer(&buffer);                      /* fetch first packet */
  while (true) {
      s.info = buffer;                              /* construct a frame for transmission */
      s.seq = next_frame_to_send;                   /* insert sequence number in frame */
      to_physical_layer(&s);                        /* send it on its way */
      start_timer(s.seq);                           /* if answer takes too long, time out */
      wait_for_event(&event);                       /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s);                  /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
              stop_timer(s.ack);                    /* turn the timer off */
              from_network_layer(&buffer);          /* get the next one to send */
              inc(next_frame_to_send);              /* invert next_frame_to_send */
          }
      }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
      wait_for_event(&event);                       /* possibilities: frame_arrival, cksum_err */
      if (event == frame_arrival) {                 /* a valid frame has arrived */
          from_physical_layer(&r);                  /* go get the newly arrived frame */
          if (r.seq == frame_expected) {            /* this is what we have been waiting for */
              to_network_layer(&r.info);            /* pass the data to the network layer */
              inc(frame_expected);                  /* next time expect the other sequence nr */
          }
          s.ack = 1 - frame_expected;               /* tell which frame is being acked */
          to_physical_layer(&s);                    /* send acknowledgement */
      }
  }
}
```

Figure 8.4 A positive acknowledgement with retransmission protocol

Protocol 3 differs from its predecessors in that both sender and receiver have a variable whose value is remembered while the data link layer is in the wait state. The sender remembers the sequence number of the next frame to send in next frame to send; the receiver remembers the sequence number of the next frame expected in frame expected. Each protocol has a short initialization phase before entering the infinite loop.

After transmitting a frame, the sender starts the timer running. If it was already running, it will be reset to allow another full timer interval. The interval should be chosen to allow enough time for the frame to get to the receiver, for the receiver to process it in the worst case, and for the acknowledgement frame to propagate back to the sender. Only when that interval has elapsed is it safe to assume that either the transmitted frame or its acknowledgement has been lost, and to send a duplicate. If the timeout interval is set too short, the sender will transmit unnecessary frames. While these extra frames will not affect the correctness of the protocol, they will hurt performance.

After transmitting a frame and starting the timer, the sender waits for something exciting to happen. Only three possibilities exist: an acknowledgement frame arrives undamaged, a damaged acknowledgement frame staggers in, or the timer expires. If a valid acknowledgement comes in, the sender fetches the next packet from its network layer and puts it in the buffer, overwriting the previous packet. It also advances the sequence number. If a damaged frame arrives or the timer expires, neither the buffer nor the sequence number is changed so that a duplicate can be sent. In all cases, the contents of the buffer (either the next packet or a duplicate) are then sent.

When a valid frame arrives at the receiver, its sequence number is checked to see if it is a duplicate. If not, it is accepted, passed to the network layer, and an acknowledgement is generated. Duplicates and damaged frames are not passed to the network layer, but they do cause the last correctly received frame to be acknowledged to signal the sender to advance to the next frame or retransmit a damaged frame.

8.4 Sliding Window Protocols

In the previous protocols, data frames were transmitted in one direction only. In most practical situations, there is a need to transmit data in both directions. One way of achieving

full-duplex data transmission is to run two instances of one of the previous protocols, each using a separate link for simplex data traffic (in different directions). Each link is then comprised of a "forward" channel (for data) and a "reverse" channel (for acknowledgements). In both cases the capacity of the reverse channel is almost entirely wasted.

A better idea is to use the same link for data in both directions. After all, in protocols 2 and 3 it was already being used to transmit frames both ways, and the reverse channel normally has the same capacity as the forward channel. In this model the data frames from A to B are intermixed with the acknowledgement frames from A to B. By looking at the kind field in the header of an incoming frame, the receiver can tell whether the frame is data or an acknowledgement.

Although interleaving data and control frames on the same link is a big improvement over having two separate physical links, yet another improvement is possible. When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes it the next packet. The acknowledgement is attached to the outgoing data frame (using the ack field in the frame header). In effect, the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as piggybacking.

The principal advantage of using piggybacking over having distinct acknowledgement frames is a better use of the available channel bandwidth. The ack field in the frame header costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum. In addition, fewer frames sent generally means a lighter processing load at the receiver. In the next protocol to be examined, the piggyback field costs only 1 bit in the frame header. It rarely costs more than a few bits.

However, piggybacking introduces a complication not present with separate acknowledgements. How long should the data link layer wait for a packet onto which to piggyback the acknowledgement? If the data link layer waits longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements. If the data link layer were an oracle and could foretell the future, it would know when the next network

layer packet was going to come in and could decide either to wait for it or send a separate acknowledgement immediately, depending on how long the projected wait was going to be. Of course, the data link layer cannot foretell the future, so it must resort to some ad hoc scheme, such as waiting a fixed number of milliseconds. If a new packet arrives quickly, the acknowledgement is piggybacked onto it. Otherwise, if no new packet has arrived by the end of this time period, the data link layer just sends a separate acknowledgement frame.

The next three protocols are bidirectional protocols that belong to a class called sliding window protocols. The three differ among themselves in terms of efficiency, complexity, and buffer requirements, as discussed later. In these, as in all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually 2n – 1 so the sequence number fits exactly in an n-bit field. The stop-and-wait sliding window protocol uses n = 1, restricting the sequence numbers to 0 and 1, but more sophisticated versions can use an arbitrary n.

The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window. Similarly, the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept. The sender's window and the receiver's window need not have the same lower and upper limits or even have the same size. In some protocols they are fixed in size, but in others they can grow or shrink over the course of time as frames are sent and received.

Although these protocols give the data link layer more freedom about the order in which it may send and receive frames, we have definitely not dropped the requirement that the protocol must deliver packets to the destination network layer in the same order they were passed to the data link layer on the sending machine. Nor have we changed the requirement that the physical communication channel is "wire-like," that is, it must deliver all frames in the order sent.

The sequence numbers within the sender's window represent frames that have been sent or can be sent but are as yet not acknowledged. Whenever a new packet arrives from the network layer, it is given the next highest sequence number, and the upper edge of the window is advanced by one. When an acknowledgement comes in, the lower edge is advanced by one.

In this way the window continuously maintains a list of unacknowledged frames. Figure 8.5 shows an example.

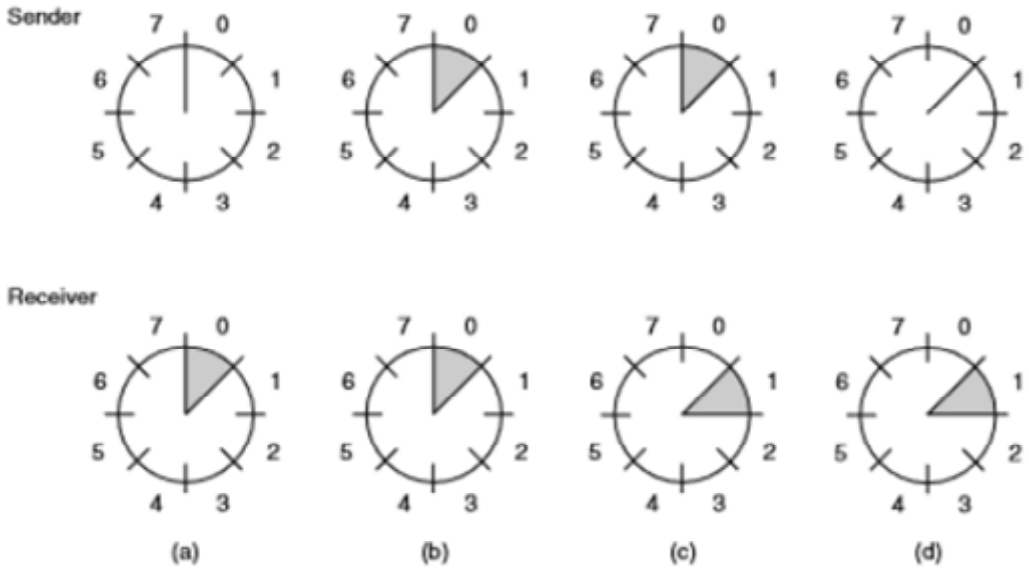


Figure 8.5 A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

Since frames currently within the sender's window may ultimately be lost or damaged in transit, the sender must keep all of these frames in its memory for possible retransmission. Thus, if the maximum window size is n, the sender needs n buffers to hold the unacknowledged frames. If the window ever grows to its maximum size, the sending data link layer must forcibly shut off the network layer until another buffer becomes free.

The receiving data link layer's window corresponds to the frames it may accept. Any frame falling within the window is put in the receiver's buffer. When a frame whose sequence number is equal to the lower edge of the window is received, it is passed to the network layer and the window is rotated by one. Any frame falling outside the window is discarded. In all of these cases, a subsequent acknowledgement is generated so that the sender may work out how to proceed. Note that a window size of 1 means that the data link layer only accepts frames in order, but for larger windows this is not so. The network layer, in contrast, is always fed data in the proper order, regardless of the data link layer's window size.

Figure 8.5 shows an example with a maximum window size of 1. Initially, no frames are outstanding, so the lower and upper edges of the sender's window are equal, but as time goes on, the situation progresses as shown. Unlike the sender's window, the receiver's window always remains at its initial size, rotating as the next frame is accepted and delivered to the network layer.

8.4.1 A One-Bit Sliding Window Protocol

Before tackling the general case, let us examine a sliding window protocol with a window size of 1. Such a protocol uses stop-and-wait since the sender transmits a frame and waits for its acknowledgement before sending the next one.

Figure 8.6 depicts such a protocol. Like the others, it starts out by defining some variables. Next frame to send tells which frame the sender is trying to send. Similarly, frame expected tells which frame the receiver is expecting. In both cases, 0 and 1 are the only possibilities.

```
/* Protocol 4 (Sliding wind                                          ol 4 */

#define MAX_SEQ 1
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;                  /* 0 or 1 only */
  seq_nr frame_expected;                      /* 0 or 1 only */
  frame r, s;                                 /* scratch variables */
  packet buffer;                              /* current packet being sent */
  event_type event;
  next_frame_to_send = 0;                     /* next frame on the outbound stream */
  frame_expected = 0;                         /* frame expected next */
  from_network_layer(&buffer);                /* fetch a packet from the network layer */
  s.info = buffer;                            /* prepare to send the initial frame */
  s.seq = next_frame_to_send;                 /* insert sequence number into frame */
  s.ack = 1 - frame_expected;                 /* piggybacked ack */
  to_physical_layer(&s);                      /* transmit the frame */
  start_timer(s.seq);                         /* start the timer running */
  while (true) {
      wait_for_event(&event);                 /* frame_arrival, cksum_err, or timeout */
      if (event == frame_arrival) {           /* a frame has arrived undamaged */
          from_physical_layer(&r);            /* go get it */
          if (r.seq == frame_expected) {      /* handle inbound frame stream */
              to_network_layer(&r.info);      /* pass packet to network layer */
              inc(frame_expected);            /* invert seq number expected next */
          }
          if (r.ack == next_frame_to_send) {  /* handle outbound frame stream */
              stop_timer(r.ack);              /* turn the timer off */
              from_network_layer(&buffer);    /* fetch new pkt from network layer */
              inc(next_frame_to_send);        /* invert sender's sequence number */
          }
      }
      s.info = buffer;                        /* construct outbound frame */
      s.seq = next_frame_to_send;             /* insert sequence number into it */
      s.ack = 1 - frame_expected;             /* seq number of last received frame */
      to_physical_layer(&s);                  /* transmit a frame */
      start_timer(s.seq);                     /* start the timer running */
  }
}
```

Figure 8.6 A 1-bit sliding window protocol

Under normal circumstances, one of the two data link layers goes first and transmits the first frame. In other words, only one of the data link layer programs should contain the to physical layer and start timer procedure calls outside the main loop. The starting machine fetches the first packet from its network layer, builds a frame from it, and sends it. When this (or any) frame arrives, the receiving data link layer checks to see if it is a duplicate, just as in protocol 3. If the frame is the one expected, it is passed to the network layer and the receiver's window is slid up.

The acknowledgement field contains the number of the last frame received without error. If this number agrees with the sequence number of the frame the sender is trying to send, the sender knows it is done with the frame stored in buffer and can fetch the next packet from its network layer. If the sequence number disagrees, it must continue trying to send the same frame. Whenever a frame is received, a frame is also sent back.

Now let us examine protocol 4 to see how resilient it is to pathological scenarios. Assume that computer A is trying to send its frame 0 to computer B and that B is trying to send its frame 0 to A. Suppose that A sends a frame to B, but A's timeout interval is a little too short. Consequently, A may time out repeatedly, sending a series of identical frames, all with seq = 0 and ack = 1.

When the first valid frame arrives at computer B, it will be accepted and frame expected will be set to a value of 1. All the subsequent frames received will be rejected because B is now expecting frames with sequence number 1, not 0. Furthermore, since all the duplicates will have ack = 1 and B is still waiting for an acknowledgement of 0, B will not go and fetch a new packet from its network layer.

After every rejected duplicate comes in, B will send A a frame containing seq = 0 and ack = 0. Eventually, one of these will arrive correctly at A, causing A to begin sending the next packet. No combination of lost frames or premature timeouts can cause the protocol to deliver duplicate packets to either network layer, to skip a packet, or to deadlock. The protocol is correct.

However, to show how subtle protocol interactions can be, we note that a peculiar situation arises if both sides simultaneously send an initial packet. This synchronization difficulty is

illustrated by Fig. 8.7. In part (a), the normal operation of the protocol is shown. In (b) the peculiarity is illustrated. If B waits for A's first frame before sending one of its own, the sequence is as shown in (a), and every frame is accepted.

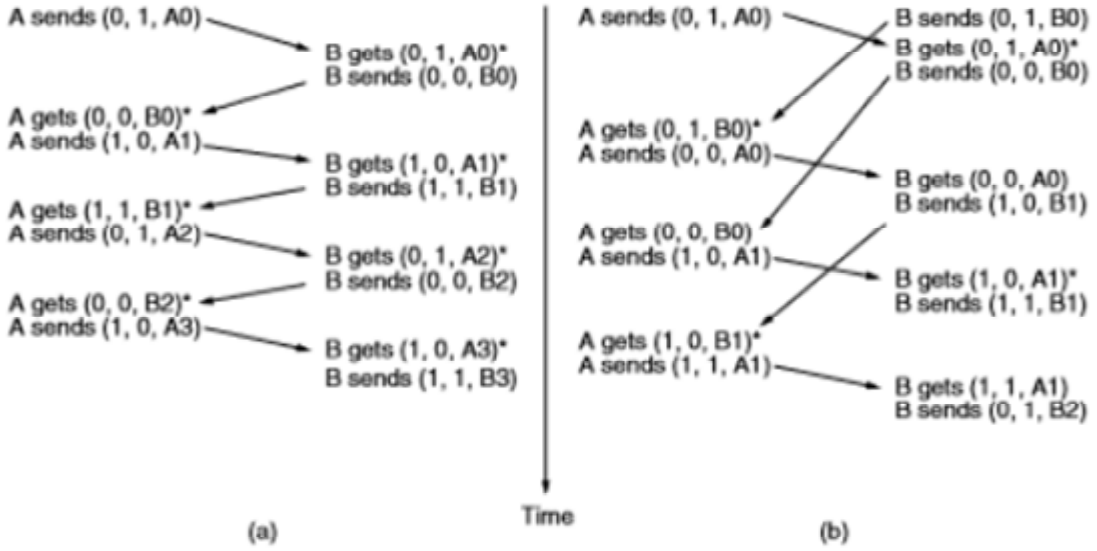


Figure 8.7 Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

However, if A and B simultaneously initiate communication, their first frames cross, and the data link layers then get into situation (b). In (a) each frame arrival brings a new packet for the network layer; there are no duplicates. In (b) half of the frames contain duplicates, even though there are no transmission errors. Similar situations can occur as a result of premature timeouts, even when one side clearly starts first. In fact, if multiple premature timeouts occur, frames may be sent three or more times, wasting valuable bandwidth.

8.4.2 A Protocol Using Go-Back-N

Until now we have made the tacit assumption that the transmission time required for a frame to arrive at the receiver plus the transmission time for the acknowledgement to come back is negligible. Sometimes this assumption is clearly false. In these situations the long round-trip time can have important implications for the efficiency of the bandwidth utilization. As an example, consider a 50-kbps satellite channel with a 500-msec round-trip propagation delay.

Let us imagine trying to use protocol 4 to send 1000-bit frames via the satellite. At t = 0 the sender starts sending the first frame. At t = 20msec the frame has been completely sent. Not until t = 270msec has the frame fully arrived at the receiver, and not until t = 520msec has the acknowledgement arrived back at the sender, under the best of circumstances (of no waiting in the receiver and a short acknowledgement frame). This means that the sender was blocked 500/520 or 96% of the time. In other words, only 4% of the available bandwidth was used. Clearly, the combination of a long transit time, high bandwidth, and short frame length is disastrous in terms of efficiency.

The problem described here can be viewed as a consequence of the rule requiring a sender to wait for an acknowledgement before sending another frame. If we relax that restriction, much better efficiency can be achieved. Basically, the solution lies in allowing the sender to transmit up to w frames before blocking, instead of just 1. With a large enough choice of w the sender will be able to continuously transmit frames since the acknowledgements will arrive for previous frames before the window becomes full, preventing the sender from blocking.

To find an appropriate value for w we need to know how many frames can fit inside the channel as they propagate from sender to receiver. This capacity is determined by the bandwidth in bits/sec multiplied by the one-way transit time, or the bandwidth-delay product of the link. We can divide this quantity by the number of bits in a frame to express it as a number of frames. Call this quantity BD. Then w should be set to 2BD + 1. Twice the bandwidth-delay is the number of frames that can be outstanding if the sender continuously sends frames when the round-trip time to receive an acknowledgement is considered. The "+1" is because an acknowledgement frame will not be sent until after a complete frame is received.

For the example link with a bandwidth of 50 kbps and a one-way transit time of 250 msec, the bandwidth-delay product is 12.5 kbit or 12.5 frames of 1000 bits each. 2BD + 1 is then 26 frames. Assume the sender begins sending frame 0 as before and sends a new frame every 20 msec. By the time it has finished sending 26 frames, at t = 520 msec, the acknowledgement for frame 0 will have just arrived. Thereafter, acknowledgements will arrive every 20 msec, so the sender will always get permission to continue just when it needs it. From then onwards, 25 or 26 unacknowledged frames will always be outstanding. Put in other terms, the sender's maximum window size is 26.

For smaller window sizes, the utilization of the link will be less than 100% since the sender will be blocked sometimes. We can write the utilization as the fraction of time that the sender is not blocked:

$$\text{link utilization} \leq \frac{w}{1 + 2BD}$$

This value is an upper bound because it does not allow for any frame processing time and treats the acknowledgement frame as having zero length, since it is usually short. The equation shows the need for having a large window w whenever the bandwidth-delay product is large. If the delay is high, the sender will rapidly exhaust its window even for a moderate bandwidth, as in the satellite example. If the bandwidth is high, even for a moderate delay the sender will exhaust its window quickly unless it has a large window (e.g., a 1-Gbps link with 1-msec delay holds 1 megabit). With stop-and-wait for which w = 1, if there is even one frame's worth of propagation delay the efficiency will be less than 50%.

This technique of keeping multiple frames in flight is an example of pipelining. Pipelining frames over an unreliable communication channel raises some serious issues. First, what happens if a frame in the middle of a long stream is damaged or lost? Large numbers of succeeding frames will arrive at the receiver before the sender even finds out that anything is wrong. When a damaged frame arrives at the receiver, it obviously should be discarded, but what should the receiver do with all the correct frames following it? Remember that the receiving data link layer is obligated to hand packets to the network layer in sequence.

Two basic approaches are available for dealing with errors in the presence of pipelining, both of which are shown in Fig. 8.8.

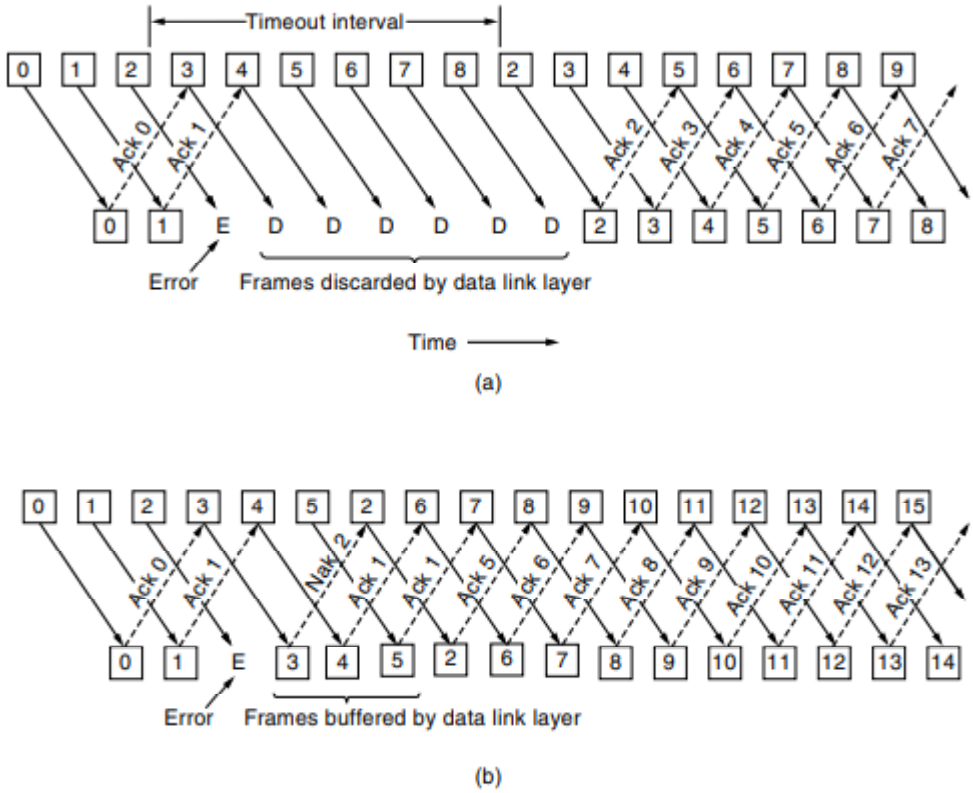


Figure 8.8 Pipelining and error recovery. Effect of an error when (a) receiver's window size is 1 and (b) receiver's window size is large

One option, called go-back-n, is for the receiver simply to discard all subsequent frames, sending no acknowledgements for the discarded frames. This strategy corresponds to a receive window of size 1. In other words, the data link layer refuses to accept any frame except the next one it must give to the network layer. If the sender's window fills up before the timer runs out, the pipeline will begin to empty. Eventually, the sender will time out and retransmit all unacknowledged frames in order, starting with the damaged or lost one. This approach can waste a lot of bandwidth if the error rate is high.

In Fig. 8.8(b) we see go-back-n for the case in which the receiver's window is large. Frames 0 and 1 are correctly received and acknowledged. Frame 2, however, is damaged or lost. The sender, unaware of this problem, continues to send frames until the timer for frame 2 expires. Then it backs up to frame 2 and starts over with it, sending 2, 3, 4, etc. all over again.

The other general strategy for handling errors when frames are pipelined is called selective repeat. When it is used, a bad frame that is received is discarded, but any good frames received after it are accepted and buffered. When the sender times out, only the oldest unacknowledged frame is retransmitted. If that frame arrives correctly, the receiver can deliver to the network layer, in sequence, all the frames it has buffered. Selective repeat corresponds to a receiver window larger than 1. This approach can require large amounts of data link layer memory if the window is large.

Selective repeat is often combined with having the receiver send a negative acknowledgement (NAK) when it detects an error, for example, when it receives a checksum error or a frame out of sequence. NAKs stimulate retransmission before the corresponding timer expires and thus improve performance.

In Fig. 8.8(b), frames 0 and 1 are again correctly received and acknowledged and frame 2 is lost. When frame 3 arrives at the receiver, the data link layer there notices that it has missed a frame, so it sends back a NAK for 2 but buffers 3. When frames 4 and 5 arrive, they, too, are buffered by the data link layer instead of being passed to the network layer. Eventually, the NAK 2 gets back to the sender, which immediately resends frame 2. When that arrives, the data link layer now has 2, 3, 4, and 5 and can pass all of them to the network layer in the correct order. It can also acknowledge all frames up to and including 5, as shown in the figure. If the NAK should get lost, eventually the sender will time out for frame 2 and send it (and only it) of its own accord, but that may be a quite a while later.

These two alternative approaches are trade-offs between efficient use of bandwidth and data link layer buffer space. Depending on which resource is scarcer, one or the other can be used. In a go-back-n protocol, the receiving data link layer only accepts frames in order; frames following an error are discarded. In this protocol, for the first time we have dropped the assumption that the network layer always has an infinite supply of packets to send. When the network layer has a packet it wants to send, it can cause a network_layer_ready event to happen. However, to enforce the flow control limit on the sender window or the number of unacknowledged frames that may be outstanding at any time, the data link layer must be able to keep the network layer from bothering it with more work. The library procedures enable_network_layerand disable_network_layer do this job.

The maximum number of frames that may be outstanding at any instant is not the same as the size of the sequence number space. For go-back-n, MAX_SEQ frames may be outstanding at any instant, even though there are MAX_SEQ + 1 distinct sequence numbers (which are 0, 1, ..., MAX_SEQ). We will see an even tighter restriction for the next protocol, selective repeat. To see why this restriction is required, consider the following scenario with MAX_SEQ = 7:

1. The sender sends frames 0 through 7.
2. A piggybacked acknowledgement for 7 comes back to the sender.
3. The sender sends another eight frames, again with sequence numbers 0 through 7.
4. Now another piggybacked acknowledgement for frame 7 comes in.

The question is this: did all eight frames belonging to the second batch arrive successfully, or did all eight get lost (counting discards following an error as lost)? In both cases the receiver would be sending frame 7 as the acknowledgement. The sender has no way of telling. For this reason the maximum number of outstanding frames must be restricted to MAX_SEQ.

Although protocol 5 does not buffer the frames arriving after an error, it does not escape the problem of buffering altogether. Since a sender may have to retransmit all the unacknowledged frames at a future time, it must hang on to all transmitted frames until it knows for sure that they have been accepted by the receiver. When an acknowledgement comes in for frame n, frames n – 1, n – 2, and so on are also automatically acknowledged. This type of acknowledgement is called a cumulative acknowledgement. This property is especially important when some of the previous acknowledgement-bearing frames were lost or garbled. Whenever any acknowledgement comes in, the data link layer checks to see if any buffers can now be released. If buffers can be released (i.e., there is some room available in the window), a previously blocked network layer can now be allowed to cause more network_layer_ready events.

For this protocol, we assume that there is always reverse traffic on which to piggyback acknowledgements. Protocol 4 does not need this assumption since it sends back one frame every time it receives a frame, even if it has already sent that frame. In the next protocol we will solve the problem of one-way traffic in an elegant way.

Because protocol 5 has multiple outstanding frames, it logically needs multiple timers, one per outstanding frame. Each frame times out independently of all the other ones. However, all of these timers can easily be simulated in software using a single hardware clock that causes interrupts periodically. The pending timeouts form a linked list, with each node of the list containing the number of clock ticks until the timer expires, the frame being timed, and a pointer to the next node.

As an illustration of how the timers could be implemented, consider the example of Fig. 8.9(a). Assume that the clock ticks once every 1 msec. Initially,the real time is 10:00:00.000; three timeouts are pending, at 10:00:00.005, 10:00:00.013, and 10:00:00.019. Every time the hardware clock ticks, the real time is updated and the tick counter at the head of the list is decremented. When the tick counter becomes zero, a timeout is caused and the node is removed from the list, as shown in Fig. 8.9(b). Although this organization requires the list to be scanned when start timer or stop timer is called, it does not require much work per tick. In protocol 5, both of these routines have been given a parameter indicating which frame is to be timed.

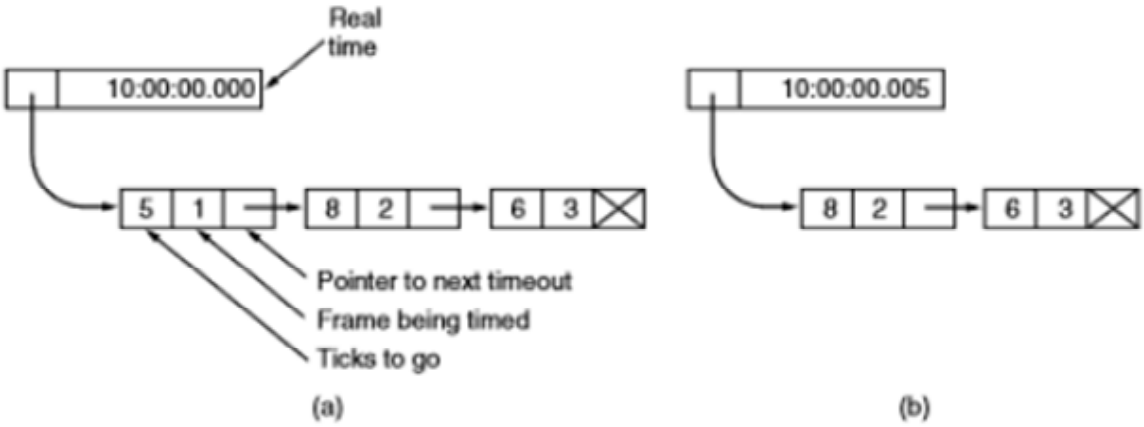


Figure 8.9 Simulation of multiple timers in software. (a) The queued timeouts. (b) The situation after the first timeout has expired

8.4.3 A Protocol Using Selective Repeat

The go-back-n protocol works well if errors are rare, but if the line is poor it wastes a lot of bandwidth on retransmitted frames. An alternative strategy, the selective repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one.

In this protocol, both sender and receiver maintain a window of outstanding and acceptable sequence numbers, respectively. The sender's window size starts out at 0 and grows to some predefined maximum. The receiver's window, in contrast, is always fixed in size and equal to the predetermined maximum. The receiver has a buffer reserved for each sequence number within its fixed window. Associated with each buffer is a bit (arrived) telling whether the buffer is full or empty. Whenever a frame arrives, its sequence number is checked by the function between to see if it falls within the window. If so and if it has not already been received, it is accepted and stored. This action is taken without regard to whether or not the frame contains the next packet expected by the network layer. Of course, it must be kept within the data link layer and not passed to the network layer until all the lower-numbered frames have already been delivered to the network layer in the correct order.

Nonsequential receive introduces further constraints on frame sequence numbers compared to protocols in which frames are only accepted in order. We can illustrate the trouble most easily with an example. Suppose that we have a 3-bit sequence number, so that the sender is permitted to transmit up to seven frames before being required to wait for an acknowledgement. Initially, the sender's and receiver's windows are as shown in Fig. 8.10(a). The sender now transmits frames 0 through 6. The receiver's window allows it to accept any frame with a sequence number between 0 and 6 inclusive. All seven frames arrive correctly, so the receiver acknowledges them and advances its window to allow receipt of 7, 0, 1, 2, 3, 4, or 5, as shown in Fig. 8.10(b). All seven buffers are marked empty.

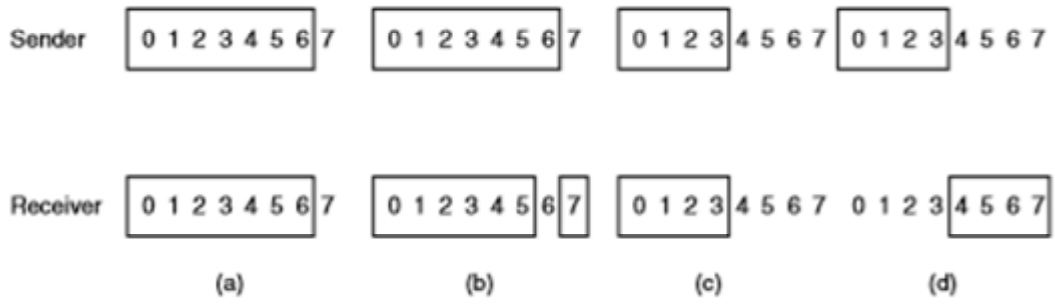


Figure 8.10(a) Initial situation with a window of size7. (b) After 7 frames have been sent and received but not acknowledged. (c) Initial situation with a window size of 4. (d) After 4 frames have been sent and received but not acknowledged.

It is at this point that disaster strikes in the form of a lightning bolt hitting the telephone pole and wiping out all the acknowledgements. The protocol should operate correctly despite this disaster. The sender eventually times out and retransmits frame 0. When this frame arrives at the receiver, a check is made to see if it falls within the receiver's window. Unfortunately, in Fig. 8.10(b) frame 0 iswithin the new window, so it is accepted as a new frame. The receiver also sends a (piggybacked) acknowledgement for frame 6, since 0 through 6 have been received.

The sender is happy to learn that all its transmitted frames did actually arrive correctly, so it advances its window and immediately sends frames 7, 0, 1, 2, 3, 4, and 5. Frame 7 will be accepted by the receiver and its packet will be passed directly to the network layer. Immediately thereafter, the receiving data link layer checks to see if it has a valid frame 0 already, discovers that it does, and passes the old buffered packet to the network layer as if it were a new packet. Consequently, the network layer gets an incorrect packet, and the protocol fails.

The essence of the problem is that after the receiver advanced its window, the new range of valid sequence numbers overlapped the old one. Consequently, the following batch of frames might be either duplicates (if all the acknowledgements were lost) or new ones (if all the acknowledgements were received). The poor receiver has no way of distinguishing these two cases.

The way out of this dilemma lies in making sure that after the receiver has advanced its window there is no overlap with the original window. To ensure that there is no overlap, the maximum window size should be at most half the range of the sequence numbers. This situation is shown in Fig. 8.10(c) and Fig. 8.10(d). With 3 bits, the sequence numbers range from 0 to 7. Only four unacknowledged frames should be outstanding at any instant. That way, if the receiver has just accepted frames 0 through 3 and advanced its window to permit acceptance of frames 4 through 7, it can unambiguously tell if subsequent frames are retransmissions (0 through 3) or new ones (4 through 7). In general, the window size for protocol 6 will be (MAX_SEQ + 1)/2.

An interesting question is: how many buffers must the receiver have? Under no conditions will it ever accept frames whose sequence numbers are below the lower edge of the window or frames whose sequence numbers are above the upper edge of the window. Consequently, the number of buffers needed is equal to the window size, not to the range of sequence numbers.

In the preceding example of a 3-bit sequence number, four buffers, numbered 0 through 3, are needed. When frame i arrives, it is put in buffer i mod 4. Notice that although i and (i + 4) mod 4 are "competing" for the same buffer, they are never within the window at the same time, because that would imply a window size of at least 5.

For the same reason, the number of timers needed is equal to the number of buffers, not to the size of the sequence space. Effectively, a timer is associated with each buffer. When the timer runs out, the contents of the buffer are retransmitted.

Protocol 6 also relaxes the implicit assumption that the channel is heavily loaded. We made this assumption in protocol 5 when we relied on frames being sent in the reverse direction on which to piggyback acknowledgements. If the reverse traffic is light, the acknowledgements may be held up for a long period of time, which can cause problems. In the extreme, if there is a lot of traffic in one direction and no traffic in the other direction, the protocol will block when the sender window reaches its maximum.

To relax this assumption, an auxiliary timer is started by start_ack_timer after an in-sequence data frame arrives. If no reverse traffic has presented itself before this timer expires, a separate acknowledgement frame is sent. An interrupt due to the auxiliary timer is called an ack_timeout event. With this arrangement, traffic flow in only one direction is possible because the lack of reverse data frames onto which acknowledgements can be piggybacked is no longer an obstacle. Only one auxiliary timer exists, and if start_ack_timer is called while the timer is running, it has no effect. The timer is not reset or extended since its purpose is to provide some minimum rate of acknowledgements.

It is essential that the timeout associated with the auxiliary timer be appreciably shorter than the timeout used for timing out data frames. This condition is required to ensure that a correctly received frame is acknowledged early enough that the frame's retransmission timer does not expire and retransmit the frame.

Protocol 6 uses a more efficient strategy than protocol 5 for dealing with errors. Whenever the receiver has reason to suspect that an error has occurred, it sends a negative acknowledgement (NAK) frame back to the sender. Such a frame is a request for retransmission

of the frame specified in the NAK. In two cases, the receiver should be suspicious: when a damaged frame arrives or a frame other than the expected one arrives (potential lost frame). To avoid making multiple requests for retransmission of the same lost frame, the receiver should keep track of whether a NAK has already been sent for a given frame. The variable no_nak in protocol 6 is true if no NAK has been sent yet for frame_expected. If the NAK gets mangled or lost, no real harm is done, since the sender will eventually time out and retransmit the missing frame anyway. If the wrong frame arrives after a NAK has been sent and lost, no_nak will be true and the auxiliary timer will be started. When it expires, an ACK will be sent to resynchronize the sender to the receiver's current status.

In some situations, the time required for a frame to propagate to the destination, be processed there, and have the acknowledgement come back is (nearly) constant. In these situations, the sender can adjust its timer to be "tight," just slightly larger than the normal time interval expected between sending a frame and receiving its acknowledgement. NAKs are not useful in this case.

However, in other situations the time can be highly variable. For example, if the reverse traffic is sporadic, the time before acknowledgement will be shorter when there is reverse traffic and longer when there is not. The sender is faced with the choice of either setting the interval to a small value (and risking unnecessary retransmissions), or setting it to a large value (and going idle for a long period after an error). Both choices waste bandwidth. In general, if the standard deviation of the acknowledgement interval is large compared to the interval itself, the timer is set "loose" to be conservative. NAKs can then appreciably speed up retransmission of lost or damaged frames.

Closely related to the matter of timeouts and NAKs is the question of determining which frame caused a timeout. In protocol 5, it is always ack_expected, because it is always the oldest. In protocol 6, there is no trivial way to determine who timed out. Suppose that frames 0 through 4 have been transmitted, meaning that the list of outstanding frames is 01234, in order from oldest to youngest. Now imagine that 0 times out, 5 (a new frame) is transmitted, 1 times out, 2 times out, and 6 (another new frame) is transmitted. At this point the list of outstanding frames is 3405126, from oldest to youngest. If all inbound traffic (i.e., acknowledgement-bearing frames) is lost for a while, the seven outstanding frames will time out in that order.

To keep the example from getting even more complicated than it already is, we have not shown the timer administration. Instead, we just assume that the variable oldest_frame is set upon timeout to indicate which frame timed out.

8.5 Summary

We examined a series of protocols that provide a reliable link layer using acknowledgements and retransmissions, or ARQ (Automatic Repeat reQuest), under more realistic assumptions. Starting from an error-free environment in which the receiver can handle any frame sent to it, we introduced flow control, followed by error control with sequence numbers and the stop-and-wait algorithm. Then we used the sliding window algorithm to allow bidirectional communication and introduce the concept of piggybacking. The last two protocols pipeline the transmission of multiple frames to prevent the sender from blocking on a link with a long propagation delay. The receiver can either discard all frames other than the next one in sequence, or buffer out-of-order frames and send negative acknowledgements for greater bandwidth efficiency. The former strategy is a go-back-n protocol, and the latter strategy is a selective repeat protocol.

8.6 Model Questions

1. Explain the One-bit Sliding window protocol.
2. Discuss in detail about the Elementary Data link protocols.
3. Illustrate: Simplex protocol for a Noisy channel.
4. Describe about the protocol using selective repeat.
5. Write short notes on: Protocol using Go-Back-N.

LESSON-9
EXAMPLE DATA LINK PROTOCOLS

Structure

9.1 Introduction

Within a single building, LANs are widely used for interconnection, but most wide-area network infrastructure is built up from point-to-point lines. Here we will examine the data link protocols found on point-to-point lines in the Internet in two common situations. The first situation is when packets are sent over SONET optical fiber links in wide-area networks. These links are widely used, for example, to connect routers in the different locations of an ISP's network.

The second situation is for ADSL links running on the local loop of the telephone network at the edge of the Internet. These links connect millions of individuals and businesses to the Internet. The Internet needs point-to-point links for these uses, as well as dial-up modems, leased lines, and cable modems, and so on. A standard protocol called PPP (Point-to-Point Protocol) is used to send packets over these links. SONET and ADSL links both apply PPP, but

in different ways.This lesson finally deals with CSMA protocols, Channel allocation methods and Multiple access protocols.

9.2 Objectives

- To explain about SONET protocol and Asymmetric Digital Subscriber Loop.
- To describe the CSMA and it's various types.
- To discuss about the Channel allocation methods
- To illustrate the Multiple Access Protocols and it's types.

9.3 Packet Over SONET

SONET is the physical layer protocol that is most commonly used over the wide-area optical fiber links that make up the backbone of communications networks, including the telephone system. It provides a bitstream that runs at a well-defined rate, for example 2.4 Gbps for an OC-48 link. This bitstream is organized as fixed-size byte payloads that recur every 125 μsec, whether or not there is user data to send.

To carry packets across these links, some framing mechanism is needed to distinguish occasional packets from the continuous bitstream in which they are transported. PPP runs on IP routers to provide this mechanism, as shown in Fig. 9.1.

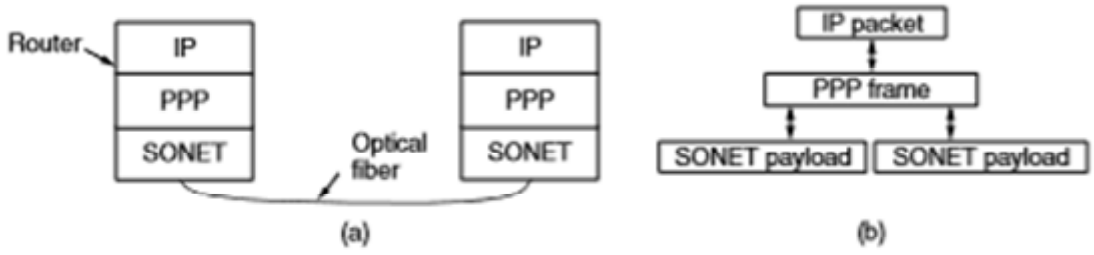


Figure 9.1 Packet over SONET. (a) A protocolstack. (b) Frame relationships

PPP improves on an earlier, simpler protocol called SLIP (Serial Line Internet Protocol) and is used to handle error detection link configuration, support multiple protocols, permit authentication, and more. With a wide set of options, PPP provides three main features:

1. A framing method that unambiguously delineates the end of one frame and the start of the next one. The frame format also handles error detection.
2. A link control protocol for bringing lines up, testing them, negotiating options, and bringing them down again gracefully when they are no longer needed. This protocol is called LCP (Link Control Protocol).
3. A way to negotiate network-layer options in a way that is independent of the network layer protocol to be used. The method chosen is to have a different NCP (Network Control Protocol) for each network layer supported.

The PPP frame format was chosen to closely resemble the frame format of HDLC (High-level Data Link Control), a widely used instance of an earlier family of protocols, since there was no need to reinvent the wheel.

The primary difference between PPP and HDLC is that PPP is byte oriented rather than bit oriented. In particular, PPP uses byte stuffing and all frames are an integral number of bytes. HDLC uses bit stuffing and allows frames of, say, 30.25 bytes.

There is a second major difference in practice, however. HDLC provides reliable transmission with a sliding window, acknowledgements, and timeouts in the manner we have studied. PPP can also provide reliable transmission in noisy environments, such as wireless networks; the exact details are defined in RFC 1663. However, this is rarely done in practice. Instead, an "unnumbered mode" is nearly always used in the Internet to provide connectionless unacknowledged service.

The PPP frame format is shown in Fig. 9.2. All PPP frames begin with the standard HDLC flag byte of 0x7E (01111110). The flag byte is stuffed if it occurs within the Payload field using the escape byte 0x7D. The following byte is the escaped byte XORed with 0x20, which flips the 5th bit. For example, 0x7D 0x5E is the escape sequence for the flag byte 0x7E. This means the start and end of frames can be searched for simply by scanning for the byte 0x7E since it will not occur elsewhere. The destuffing rule when receiving a frame is to look for 0x7D, remove it, and XOR the following byte with 0x20. Also, only one flag byte is needed between frames. Multiple flag bytes can be used to fill the link when there are no frames to be sent.

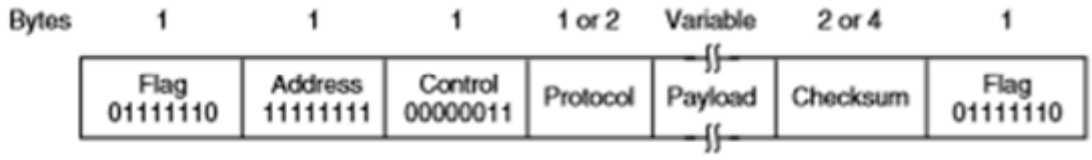


Figure 9.2 The PPP full frame format for unnumbered mode operation

After the start-of-frame flag byte comes the Address field. This field is always set to the binary value 11111111 to indicate that all stations are to accept the frame. Using this value avoids the issue of having to assign data link addresses.The Address field is followed by the Control field, the default value of which is 00000011. This value indicates an unnumbered frame. Since the Address and Control fields are always constant in the default configuration, LCP provides the necessary mechanism for the two parties to negotiate an option to omit them altogether and save 2 bytes per frame.

The fourth PPP field is the Protocol field. Its job is to tell what kind of packet is in the Payload field. Codes starting with a 0 bit are defined for IP version 4, IP version 6, and other network layer protocols that might be used, such as IPX andAppleTalk. Codes starting with a 1 bit are used for PPP configuration protocols, including LCP and a different NCP for each network layer protocol supported. The default size of the Protocol field is 2 bytes, but it can be negotiated down to 1 byte using LCP. The designers were perhaps overly cautious in thinking that someday there might be more than 256 protocols in use.

The Payload field is variable length, up to some negotiated maximum. If the length is not negotiated using LCP during line setup, a default length of 1500 bytes is used. Padding may follow the payload if it is needed. After the Payload field comes the Checksum field, which is normally 2 bytes, but a 4-byte checksum can be negotiated. The 2-byte checksum is also an industry-standard CRC.

PPP is a framing mechanism that can carry the packets of multiple protocols over many types of physical layers. To use PPP over SONET, the choices to make are spelled out in RFC 2615 (Malis and Simpson, 1999). A 4-byte checksum is used, since this is the primary means of

detecting transmission errors over the physical, link, and network layers. It is recommended that the Address, Control, and Protocol fields not be compressed, since SONET links already run at relatively high rates.

Before PPP frames can be carried over SONET lines, the PPP link must be established and configured. The phases that the link goes through when it is brought up, used, and taken down again are shown in Fig. 9.3.

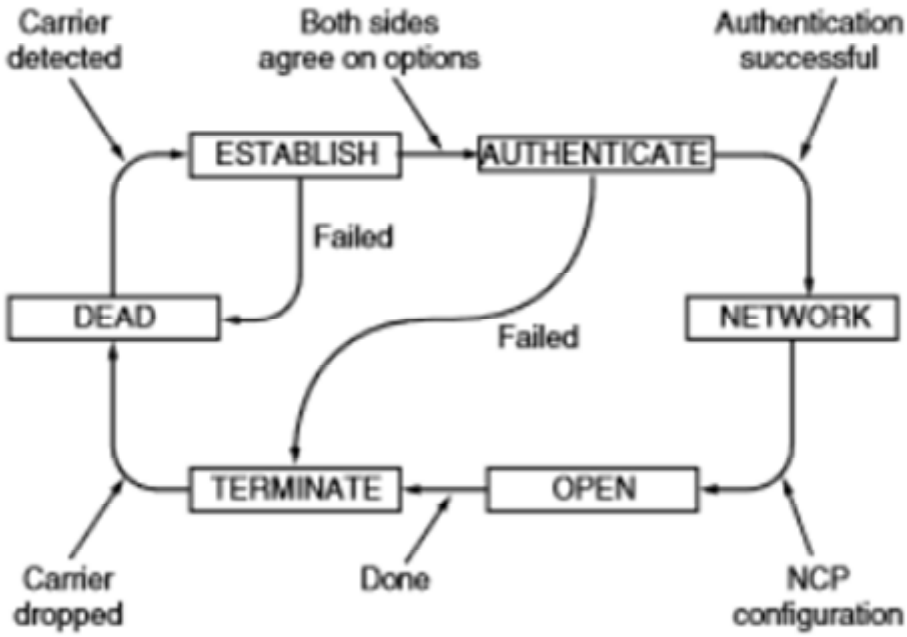


Figure 9.3 State diagram for bringing a PPP link up and down

The link starts in the DEAD state, which means that there is no connection at the physical layer. When a physical layer connection is established, the link moves to ESTABLISH. At this point, the PPP peers exchange a series of LCP packets, each carried in the Payload field of a PPP frame, to select the PPP options for the link from the possibilities mentioned above. The initiating peer proposes options, and the responding peer either accepts or rejects them, in whole or part. The responder can also make alternative proposals.

If LCP option negotiation is successful, the link reaches the AUTHENTICATE state. Now the two parties can check each other's identities, if desired. If authentication is successful, the

NETWORK state is entered and a series of NCP packets are sent to configure the network layer. It is difficult to generalize about the NCP protocols because each one is specific to some network layer protocol and allows configuration requests to be made that are specific to that protocol.For IP, for example, the assignment of IP addresses to both ends of the link is the most important possibility.

Once OPEN is reached, data transport can take place. It is in this state that IP packets are carried in PPP frames across the SONET line. When data transport is finished, the link moves into the TERMINATE state, and from there it moves back to the DEAD state when the physical layer connection is dropped.

9.4 ADSL (Asymmetric Digital Subscriber Loop)

ADSL connects millions of home subscribers to the Internet at megabit/sec rates over the same telephone local loop that is used for plain old telephone service. In this section, we will explore in more detail how packets are carried over ADSL links.

The overall picture for the protocols and devices used with ADSL is shown in Fig. 9.4. Different protocols are deployed in different networks, so we have chosen to show the most popular scenario. Inside the home, a computer such as a PC sends IP packets to the DSL modem using a link layer like Ethernet. The DSL modem then sends the IP packets over the local loop to the DSLAM using the protocols that we are about to study. At the DSLAM (or a router connected to it depending on the implementation) the IP packets are extracted and enter an ISP network so that they may reach any destination on the Internet.

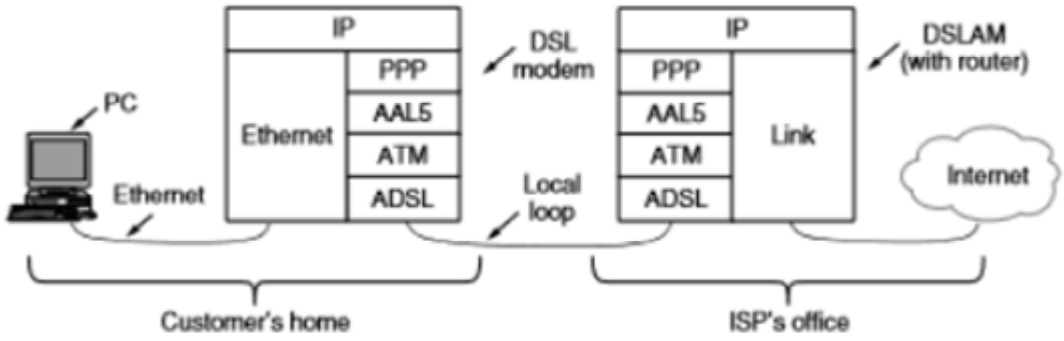


Figure 9.4 ADSL protocol stacks

The protocols shown over the ADSL link in Fig. 9.4 start at the bottom with the ADSL physical layer. They are based on a digital modulation scheme calledorthogonal frequency division multiplexing (also known as discrete multitone). Near the top of the stack, just below the IP network layer, is PPP. This protocol is the same PPP that we have just studied for packet over SONET transports. It works in the same way to establish and configure the link and carry IP packets.

In between ADSL and PPP are ATM and AAL5. ATM (Asynchronous Transfer Mode) was designed in the early 1990s and launched with incredible hype. It promised a network technology that would solve the world's telecommunications problems by merging voice, data, cable television, telegraph, carrier pigeon, tin cans connected by strings, tom toms, and everything else into an integrated system that could do everything for everyone. This did not happen. In large part, the problems of ATM were similar to those we described concerning the OSI protocols, that is, bad timing, technology, implementation, and politics. Nevertheless, ATM was much more successful than OSI. While it has not taken over the world, it remains widely used in niches including broadband access lines such as DSL, and WAN links inside telephone networks.

ATM is a link layer that is based on the transmission of fixed-length cells of information. The "Asynchronous" in its name means that the cells do not always need to be sent in the way that bits are continuously sent over synchronous lines, as in SONET. Cells only need to be sent when there is information to carry. ATM is a connection-oriented technology. Each cell carries a virtual circuit identifier in its header and devices use this identifier to forward cells along the paths of established connections.

The cells are each 53 bytes long, consisting of a 48-byte payload plus a 5-byte header. By using small cells, ATM can flexibly divide the bandwidth of a physical layer link among different users in fine slices. This ability is useful when, for example, sending both voice and data over one link without having long data packets that would cause large variations in the delay of the voice samples. The unusual choice for the cell length (e.g., compared to the more natural choice of a power of 2) is an indication of just how political the design of ATM was. The 48-byte size for the payload was a compromise to resolve a deadlock between Europe, which wanted 32-byte cells, and the U.S., which wanted 64-byte cells.

To send data over an ATM network, it needs to be mapped into a sequence of cells. This mapping is done with an ATM adaptation layer in a process called segmentation and reassembly. Several adaptation layers have been defined for different services, ranging from periodic voice samples to packet data. The main one used for packet data is AAL5 (ATM Adaptation Layer 5).

An AAL5 frame is shown in Fig. 9.5. Instead of a header, it has a trailer that gives the length and has a 4-byte CRC for error detection. Naturally, the CRC is the same one used for PPP and IEEE 802 LANs like Ethernet. Wang and Crowcroft (1992) have shown that it is strong enough to detect nontraditional errors such as cell reordering. As well as a payload, the AAL5 frame has padding. This rounds out the overall length to be a multiple of 48 bytes so that the frame can be evenly divided into cells. No addresses are needed on the frame as the virtual circuit identifier carried in each cell will get it to the right destination.

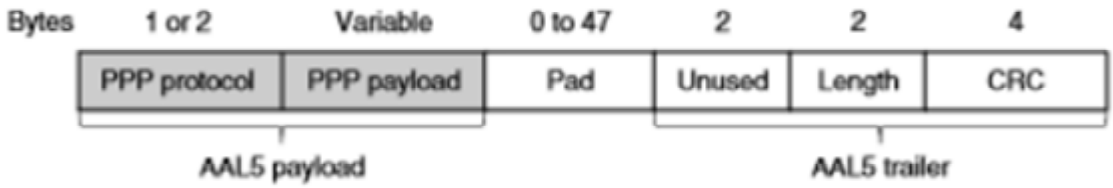


Figure 9.5 AAL5 frame carrying PPP data

Now that we have described ATM, we have only to describe how PPP makes use of ATM in the case of ADSL. It is done with yet another standard called PPPoA (PPP over ATM). This standard is not really a protocol but more a specification of how to work with both PPP and AAL5 frames. It is described in RFC 2364 (Gross et al., 1998).

Only the PPP protocol and payload fields are placed in the AAL5 payload, as shown in Fig. 9.5. The protocol field indicates to the DSLAM at the far end whether the payload is an IP packet or a packet from another protocol such as LCP. The far end knows that the cells contain PPP information because an ATM virtual circuit is set up for this purpose.

Within the AAL5 frame, PPP framing is not needed as it would serve no purpose; ATM and AAL5 already provide the framing. More framing would be worthless. The PPP CRC is also not needed because AAL5 already includes the very same CRC. This error detection mechanism supplements the ADSL physical layer coding of a Reed-Solomon code for error correction and

a 1-byte CRC for the detection of any remaining errors not otherwise caught. This scheme has a much more sophisticated error-recovery mechanism than when packets are sent over a SONET line because ADSL is a much noisier channel.

9.5 The Medium Access Control Sublayer

In any broadcast network, the key issue is how to determine who gets to use the channel when there is competition for it. To make this point, consider a conference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others. It is very likely that when one of them stops speaking, two or more will start talking at once, leading to chaos. In a face-to-face meeting, chaos is avoided by external means. For example, at a meeting, people raise their hands to request permission to speak. When only a single channel is available, it is much harder to determine who should go next. Many protocols for solving the problem are known. They form the contents of this chapter. In the literature, broadcast channels are sometimes referred to as multiaccess channels or random access channels.

The protocols used to determine who goes next on a multiaccess channel belong to a sublayer of the data link layer called the MAC (Medium Access Control) sublayer. The MAC sublayer is especially important in LANs, particularly wireless ones because wireless is naturally a broadcast channel. WANs, in contrast, use point-to-point links, except for satellite networks. Because multiaccess channels and LANs are so closely related, in this section we will discuss LANs ingeneral, including a few issues that are not strictly part of the MAC sublayer, but the main subject here will be control of the channel.

Technically, the MAC sublayer is the bottom part of the data link layer, so logically we should have studied it before examining all the point-to-point protocols in Chap. 3. Nevertheless, for most people, it is easier to understand protocols involving multiple parties after two-party protocols are well understood. For that reason we have deviated slightly from a strict bottom-up order of presentation.

9.6 The Channel Allocation Problem

The central theme of this section is how to allocate a single broadcast channel among competing users. The channel might be a portion of the wireless spectrum in a geographic region, or a single wire or optical fiber to which multiple nodes are connected. It does not matter. In both cases, the channel connects each user to all other users and any user who makes full use of the channel interferes with other users who also wish to use the channel.

We will first look at the shortcomings of static allocation schemes for bursty traffic. Then, we will lay out the key assumptions used to model the dynamic schemes that we examine in the following sections.

9.6.1 Static Channel Allocation

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is to chop up its capacity by using one of the multiplexing schemes such as FDM (Frequency Division Multiplexing). If there are N users, the bandwidth is divided into N equal-sized portions, with each user being assigned one portion. Since each user has a private frequency band, there is now no interference among users. When there is only a small and constant number of users, each of which has a steady stream or a heavy load of traffic, this division is a simple and efficient allocation mechanism. A wireless example is FM radio stations. Each station gets a portion of the FM band and uses it most of the time to broadcast its signal.

However, when the number of senders is large and varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into N regions and fewer than N users are currently interested in communicating, a large piece of valuable spectrum will be wasted. And if more than N users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

Even assuming that the number of users could somehow be held constant at N, dividing the single available channel into some number of static subchannels isinherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They

are not using it, and no one else is allowed to use it either. A static allocation is a poor fit to most computer systems, in which data traffic is extremely bursty, often with peak traffic to mean traffic ratios of 1000:1. Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen with a simple queueing theory calculation. Let us start by finding the mean time delay, T, to send a frame onto a channel of capacity C bps. We assume that the frames arrive randomly with an average arrival rate of λ frames/sec, and that the frames vary in length with an average length of $1/\mu$ bits. With these parameters, the service rate of the channel is μC frames/sec. A standard queueing theory result is

$$T = \frac{1}{\mu C - \lambda}$$

In our example, if C is 100 Mbps, the mean frame length, $1/\mu$, is 10,000 bits, and the frame arrival rate, λ, is 5000 frames/sec, then T = 200 μsec. Note that if we ignored the queueing delay and just asked how long it takes to send a 10,000bit frame on a 100-Mbps network, we would get the (incorrect) answer of 100 μsec. That result only holds when there is no contention for the channel.

Now let us divide the single channel into N independent subchannels, each with capacity C/Nbps. The mean input rate on each of the subchannels will now be λ/N. Recomputing T, we get

$$T_N = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT$$

The mean delay for the divided channel is N times worse than if all the frames were somehow magically arranged orderly in a big central queue. This same result says that a bank lobby full of ATM machines is better off having a single queue feeding all the machines than a separate queue in front of each machine.

Precisely the same arguments that apply to FDM also apply to other ways of statically dividing the channel. If we were to use time division multiplexing (TDM) and allocate each user every Nth time slot, if a user does not use the allocated slot, it would just lie fallow. The same would hold if we split up the networks physically. Using our previous example again, if we were to replace the 100-Mbps network with 10 networks of 10 Mbps each and statically allocate each user to one of them, the mean delay would jump from 200 μsec to 2 μsec. Since none of the traditional static channel allocation methods work well at all with bursty traffic, we will now explore dynamic methods.

9.6.2 Assumptions for Dynamic Channel Allocation

Before we get to the first of the many channel allocation methods in this chapter, it is worthwhile to carefully formulate the allocation problem. Underlying all the work done in this area are the following five key assumptions:

1. Independent Traffic. The model consists of N independent stations (e.g., computers, telephones), each with a program or user that generates frames for transmission. The expected number of frames generated in an interval of length Δt is $\Delta\lambda t$, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

2. Single Channel. A single channel is available for all communication. All stations can transmit on it and all can receive from it. The stations are assumed to be equally capable, though protocols may assign them different roles (e.g., priorities).

3. Observable Collisions. If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a collision. All stations can detect that a collision has occurred. A collided frame must be transmitted again later. No errors other than those generated by collisions occur.

4. Continuous or Slotted Time. Time may be assumed continuous, in which case frame transmission can begin at any instant. Alternatively, time may be slotted or divided into discrete intervals (called slots). Frame transmissions must then begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.

5. Carrier Sense or No Carrier Sense. With the carrier sense assumption, stations can tell if the channel is in use before trying to use it. No station will attempt to use the channel while it is sensed as busy. If there is no carrier sense, stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

Some discussion of these assumptions is in order. The first one says that frame arrivals are independent, both across stations and at a particular station, and that frames are generated unpredictably but at a constant rate. Actually, this assumption is not a particularly good model of network traffic, as it is well known that packets come in bursts over a range of time scales.

The single-channel assumption is the heart of the model. No external ways to communicate exist. Stations cannot raise their hands to request that the teacher call on them, so we will have to come up with better solutions. The remaining three assumptions depend on the engineering of the system, and we will say which assumptions hold when we examine a particular protocol.

The collision assumption is basic. Stations need some way to detect collisions if they are to retransmit frames rather than let them be lost. For wired channels, node hardware can be designed to detect collisions when they occur. The stations can then terminate their transmissions prematurely to avoid wasting capacity. This detection is much harder for wireless channels, so collisions are usually inferred after the fact by the lack of an expected acknowledgement frame. It is also possible for some frames involved in a collision to be successfully received, depending on the details of the signals and the receiving hardware. However, this situation is not the common case, so we will assume that all frames involved in a collision are lost. We will also see protocols that are designed to prevent collisions from occurring in the first place.

The reason for the two alternative assumptions about time is that slotted time can be used to improve performance. However, it requires the stations to follow a master clock or synchronize their actions with each other to divide time into discrete intervals. Hence, it is not always available.

Similarly, a network may have carrier sensing or not have it. Wired networks will generally have carrier sense. Wireless networks cannot always use it effectively because not every station

may be within radio range of every other station. Similarly, carrier sense will not be available in other settings in which a station cannot communicate directly with other stations, for example a cable modem in which stations must communicate via the cable headend. Note that the word "carrier" in this sense refers to a signal on the channel and has nothing to do with the common carriers (e.g., telephone companies) that date back to the days of the Pony Express.

To avoid any misunderstanding, it is worth noting that no multiaccess protocol guarantees reliable delivery. Even in the absence of collisions, the receiver may have copied some of the frame incorrectly for various reasons. Other parts of the link layer or higher layers provide reliability.

9.10 Multiple Access Protocols

Many algorithms for allocating a multiple access channel are known. In the following sections, we will study a small sample of the more interesting ones and give some examples of how they are commonly used in practice.

9.7.1 ALOHA

The researcher Norman Abramson and his colleagues at the University of Hawaii were trying to connect users on remote islands to the main computer in Honolulu. Stringing their own cables under the Pacific Ocean was not in the cards, so they looked for a different solution.

The one they found used short-range radios, with each user terminal sharing the same upstream frequency to send frames to the central computer. It included a simple and elegant method to solve the channel allocation problem. Although Abramson's work, called the ALOHA system, used groundbased radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel. We will discuss two versions of ALOHA here: pure and slotted. They differ with respect to whether time is continuous, as in the pure version, or divided into discrete slots into which all frames must fit.

Pure ALOHA

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged.

Senders need some way to find out if this is the case. In the ALOHA system, after each station has sent its frame to the central computer, this computer rebroadcasts the frame to all of the stations. A sending station can thus listen for the broadcast from the hub to see if its frame has gotten through. In other systems, such as wired LANs, the sender might be able to listen for collisions while transmitting.

If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that can lead to conflicts are known as contention systems.

A sketch of frame generation in an ALOHA system is given in Fig. 9.6. We have made the frames all the same length because the throughput of ALOHA systems is maximized by having a uniform frame size rather than by allowing variable-length frames.

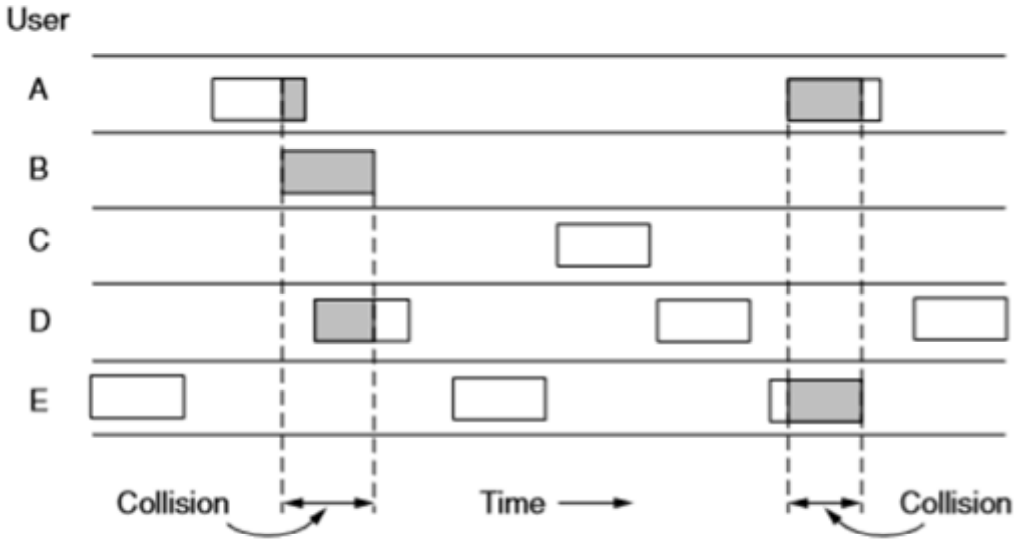


Figure 9.6 In pure ALOHA, frames are transmitted at completely arbitrary times

Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame that has almost finished, both frames will be totally destroyed (i.e., have incorrect checksums) and both will have to be retransmitted later. The checksum does not (and should not) distinguish between a total loss and a near miss.

Slotted ALOHA

Soon after ALOHA came onto the scene, Roberts (1972) published a method for doubling the capacity of an ALOHA system. His proposal was to divide time into discrete intervals called slots, each interval corresponding to one frame. This approach requires the users to agree on slot boundaries. One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

In Roberts' method, which has come to be known as slotted ALOHA-in contrast to Abramson's pure ALOHA-a station is not permitted to send whenever the user types a line. Instead, it is required to wait for the beginning of the next slot. Thus, the continuous time ALOHA is turned into a discrete time one. This halves the vulnerable period.

9.7.2 Carrier Sense Multiple Access Protocols (CSMA)

Protocols in which stations listen for a carrier (i.e., a transmission) and act accordingly are called carrier sense protocols. A number of them have been proposed and now we will look at several versions of carrier sense protocols.

Persistent and Nonpersistent CSMA

The first carrier sense protocol that we will study here is called 1-persistent CSMA (Carrier Sense Multiple Access). That is a bit of a mouthful for the simplest CSMA scheme. When a station has data to send, it first listens to the channel to see if anyone else is transmitting at that moment. If the channel is idle, the stations sends its data. Otherwise, if the channel is busy, the station just waits until it becomes idle. Then the station transmits a frame. If a collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1-persistent because the station transmits with a probability of 1 when it finds the channel idle.

If two stations become ready in the middle of a third station's transmission, both will wait politely until the transmission ends, and then both will begin transmitting exactly simultaneously, resulting in a collision. If they were not so impatient, there would be fewer collisions. More subtly, the propagation delay has an important effect on collisions. There is a chance that just after a station begins sending, another station will become ready to send and sense the channel.

If the first station's signal has not yet reached the second one, the latter will sense an idle channel and will also begin sending, resulting in a collision. This chance depends on the number of frames that fit on the channel, or the bandwidth-delay product of the channel.

A second carrier sense protocol is nonpersistent CSMA. In this protocol, a conscious attempt is made to be less greedy than in the previous one. As before, a station senses the channel when it wants to send a frame, and if no one else is sending, the station begins doing so itself. However, if the channel is already in use, the station does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm. Consequently, this algorithm leads to better channel utilization but longer delays than 1-persistent CSMA.

The last protocol is p-persistent CSMA. It applies to slotted channels and works as follows. When a station becomes ready to send, it senses the channel. If it is idle, it transmits with a probability p. With a probability $q = 1 – p$, it defers until the next slot. If that slot is also idle, it either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another station has begun transmitting.

CSMA with Collision Detection

Persistent and nonpersistent CSMA protocols are definitely an improvement over ALOHA because they ensure that no station begins to transmit while the channel is busy. However, if two stations sense the channel to be idle and begin transmitting simultaneously, their signals will still collide. Another improvement is for the stations to quickly detect the collision and abruptly stop transmitting, since they are irretrievably garbled anyway. This strategy saves time and bandwidth.

This protocol, known as CSMA/CD (CSMA with Collision Detection), is the basis of the classic Ethernet LAN, so it is worth devoting some time to looking at it in detail. It is important to realize that collision detection is an analog process. The station's hardware must listen to the channel while it is transmitting. If the signal it reads back is different from the signal it is putting out, it knows that a collision is occurring. The implications are that a received signal must not be tiny compared to the transmitted signal and that the modulation must be chosen to allow collisions to be detected.

9.7.3 Collision-Free Protocols

Although collisions do not occur with CSMA/CD once a station has unambiguously captured the channel, they can still occur during the contention period. These collisions adversely affect the system performance, especially when the bandwidth-delay product is large, such as when the cable is long and the frames are short. Not only do collisions reduce bandwidth, but they make the time to send a frame variable, which is not a good fit for real-time traffic such as voice over IP. CSMA/CD is also not universally applicable.

In this section, we will examine some protocols that resolve the contention for the channel without any collisions at all, not even during the contention period. Most of these protocols are not currently used in major systems, but in a rapidly changing field, having some protocols with excellent properties available for future systems is often a good thing.

A Bit-Map Protocol

In our first collision-free protocol, the basic bit-map method, each contention period consists of exactly N slots. If station 0 has a frame to send, it transmits a 1 bit during the slot 0. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1 bit during slot 1, but only if it has a frame queued. In general, station j may announce that it has a frame to send by inserting a 1 bit into slot j. After all N slots have passed by, each station has complete knowledge of which stations wish to transmit. At that point, they begin transmitting frames in numerical order.

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another N-bit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again.Protocols like this in which the desire to transmit is broadcast before the actual transmission are called reservation protocols because they reserve channel ownership in advance and prevent collisions.

Token Passing

The essence of the bit-map protocol is that it lets every station transmit a frame in turn in a predefined order. Another way to accomplish the same thing is to pass a small message called a token from one station to the next in the same predefined order. The token represents permission to send. If a station has a frame queued for transmission when it receives the token, it can send that frame before it passes the token to the next station. If it has no queued frame, it simply passes the token.

In a token ring protocol, the topology of the network is used to define the order in which stations send. The stations are connected one to the next in a single ring. Passing the token to the next station then simply consists of receiving the token in from one direction and transmitting it out in the other direction, as seen in Fig. 9.7. Frames are also transmitted in the direction of the token. This way they will circulate around the ring and reach whichever station is the destination. However, to stop the frame circulating indefinitely (like the token), some station needsto remove it from the ring. This station may be either the one that originally sent the frame, after it has gone through a complete cycle, or the station that was the intended recipient of the frame.

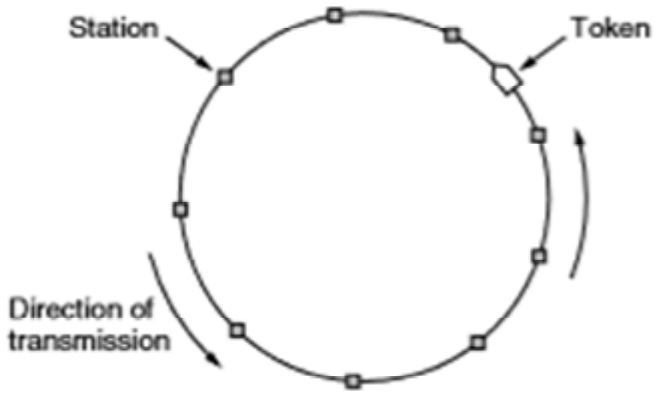


Figure 9.7 Token ring

Note that we do not need a physical ring to implement token passing. The channel connecting the stations might instead be a single long bus. Each station then uses the bus to send the token to the next station in the predefined sequence. Possession of the token allows a station to use the bus to send one frame, as before. This protocol is called token bus.

Binary Countdown

A problem with the basic bit-map protocol, and by extension token passing, is that the overhead is 1 bit per station, so it does not scale well to networks with thousands of stations. We can do better than that by using binary station addresses with a channel that combines transmissions. A station wanting to use thechannel now broadcasts its address as a binary bit string, starting with the highorder bit. All addresses are assumed to be the same length. The bits in each address position from different stations are BOOLEAN ORed together by the channel when they are sent at the same time. We will call this protocol binary countdown.

To avoid conflicts, an arbitration rule must be applied: as soon as a station sees that a high-order bit position that is 0 in its address has been overwritten with a 1, it gives up. It has the property that higher-numbered stations have a higher priority than lower-numbered stations, which may be either good or bad, depending on the context.Binary countdown is an example of a simple, elegant, and efficient protocol that is waiting to be rediscovered.

9.7.4 Limited-Contention Protocols

It would be nice if we could combine the best properties of the contention and collision-free protocols, arriving at a new protocol that used contention at low load to provide low delay, but used a collision-free technique at high load to provide good channel efficiency. Such protocols, which we will call limited-contention protocols.

Up to now, the only contention protocols we have studied have been symmetric. That is, each station attempts to acquire the channel with some probability, p, with all stations using the same p. Interestingly enough, the overall system performance can sometimes be improved by using a protocol that assigns different probabilities to different stations.

The limited-contention protocols first divide the stations into groups. Only the members of group 0 are permittedto compete for slot 0. If one of them succeeds, it acquires the channel and transmits its frame. If the slot lies fallow or if there is a collision, the members of group 1 contend for slot 1, etc. By making an appropriate division of stations into groups, the amount of contention for each slot can be reduced, thus operating each slot.

The Adaptive Tree Walk Protocol

It is convenient to think of the stations as the leaves of a binary tree. In the first contention slot following a successful frame transmission, slot 0, all stations are permitted to try to acquire the channel. If one of them does so, fine. If there is a collision, then during slot 1 only those stations falling under node 2 in the tree may compete. If one of them acquires the channel, the slot following the frame is reserved for those stations under node 3. If, on the other hand, two or more stations under node 2 want to transmit, there will be a collision during slot 1, in which case it is node 4's turn during slot 2.

In essence, if a collision occurs during slot 0, the entire tree is searched, depth first, to locate all ready stations. Each bit slot is associated with some particular node in the tree. If a collision occurs, the search continues recursively with the node's left and right children. If a bit slot is idle or if only one station transmits in it, the searching of its node can stop because all ready stations have been located. When the load on the system is heavy, it is hardly worth the effort to dedicate slot 0 to node 1 because that makes sense only in the unlikely event that precisely one station has a frame to send.

9.7.5 Wireless LAN Protocols

A system of laptop computers that communicate by radio can be regarded as a wireless LAN. Such a LAN is an example of a broadcast channel. It also has somewhat different properties than a wired LAN, which leads to different MAC protocols.

A common configuration for a wireless LAN is an office building with access points (APs) strategically placed around the building. The APs are wired together using copper or fiber and provide connectivity to the stations that talk to them. If the transmission power of the APs and laptops is adjusted to have a range of tens of meters, nearby rooms become like a single cell and the entire building becomes like the cellular telephony systems, except that each cell only has one channel. This channel is shared by all the stations in the cell, including the AP. It typically provides megabit/sec bandwidths, up to 600 Mbps.

There is an even more important difference between wireless LANs and wired LANs. A station on a wireless LAN may not be able to transmit frames to or receive frames from all other

stations because of the limited radio range of the stations. In wired LANs, when one station sends a frame, all other stations receive it. The absence of this property in wireless LANs causes a variety of complications.

9.8 Summary

The Internet uses PPP as the main data link protocol over point-to-point lines. It provides a connectionless unacknowledged service, using flag bytes to delimit frames and a CRC for error detection. It is used to carry packets across a range of links, including SONET links in wide-area networks and ADSL links for the home.

9.9 Model Questions

1. Write short notes on Wireless LAN.
2. Describe the various Multiple access protocols.
3. Explain Collision Free Protocols.
4. Illustrate Limited Contention Protocols.
5. What is CSMA and explain about it's types.

LESSON-10
THE NETWORK LAYER

Structure

10.1 Introduction

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other. Thus, the network layer is the lowest layer that deals with end-to-end transmission.

To achieve its goals, the network layer must know about the topology of the network (i.e., the set of all routers and links) and choose appropriate paths through it, even for large networks. It must also take care when choosing routes to avoid overloading some of the communication lines and routers while leaving others idle. Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them.

10.2 Objectives

- To explain the design issues of Network layer.
- To illustrate the Routing and various routing algorithms.

10.3 Network Layer Design Issues

In the following sections, we will give an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the network.

10.3.1 Store-and-Forward Packet Switching

Before starting to explain the details of the network layer, it is worth restating the context in which the network layer protocols operate. This context can be seen in Fig. 10.1. The major components of the network are the ISP's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host H1 is directly connected to one of the ISP's routers, A, perhaps as a home computer that is plugged into a DSL modem. In contrast, H2 is on a LAN, which might be an office Ethernet, with a router, F, owned and operated by the customer. This router has a leased line to the ISP's equipment. The router F is present outside the oval because it does not belong to the ISP.

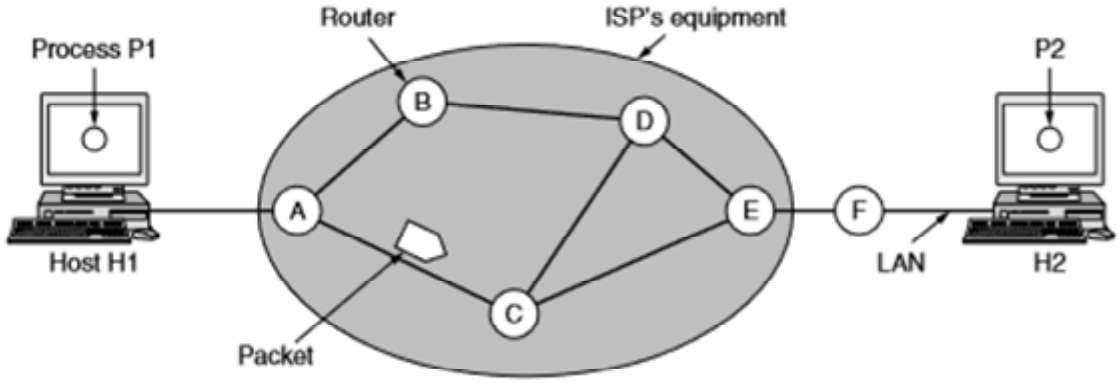


Figure 10.1 The environment of the network layer protocols

This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.

10.3.2 Services Provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

10.3.3 Implementation of Connectionless Service

Two different organizations are possible, depending on the type of service offered. If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called datagrams and the network is called a datagram network. If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a VC (virtual circuit) and the network is called a virtual-circuit network.

Let us now see how a datagram network works. Suppose that the process P1 in Fig. 10.2 has a long message for P2. It hands the message to the transport layer, with instructions to deliver it to process P2 on host H2. The transport layer code runs on H1, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer.

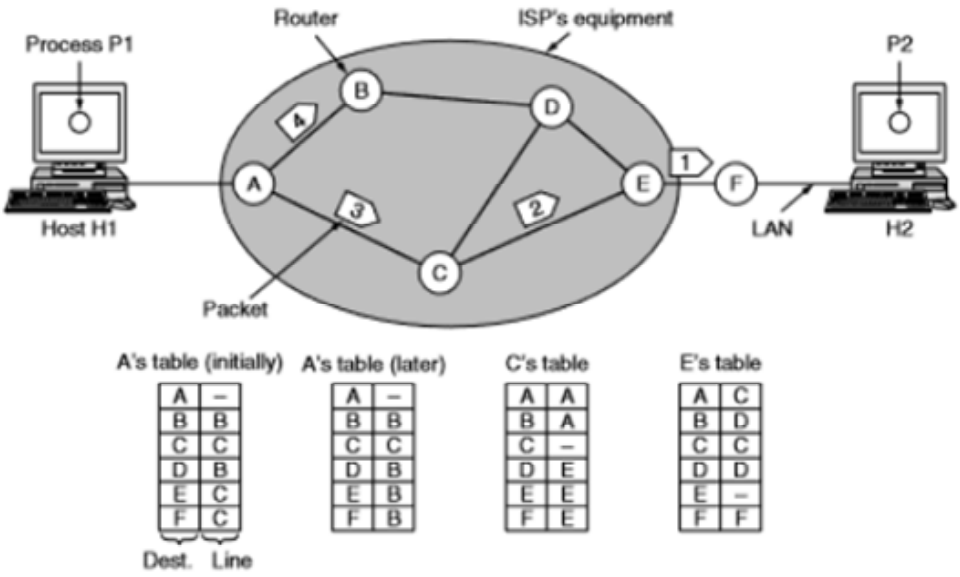


Figure 10.2 Routing within a datagram network

The algorithm that manages the tables and makes the routing decisions is called the routing algorithm. IP (Internet Protocol), which is the basis for the entire Internet, is the dominant example of a connectionless network service. Each packet carries a destination IP address that routers use to individually forward each packet. The addresses are 32 bits in IPv4 packets and 128 bits in IPv6 packets.

10.3.4 Implementation of Connection-Oriented Service

For connection-oriented service, we need a virtual-circuit network. Let us see how that works. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in Fig. 10.2. Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works. When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.

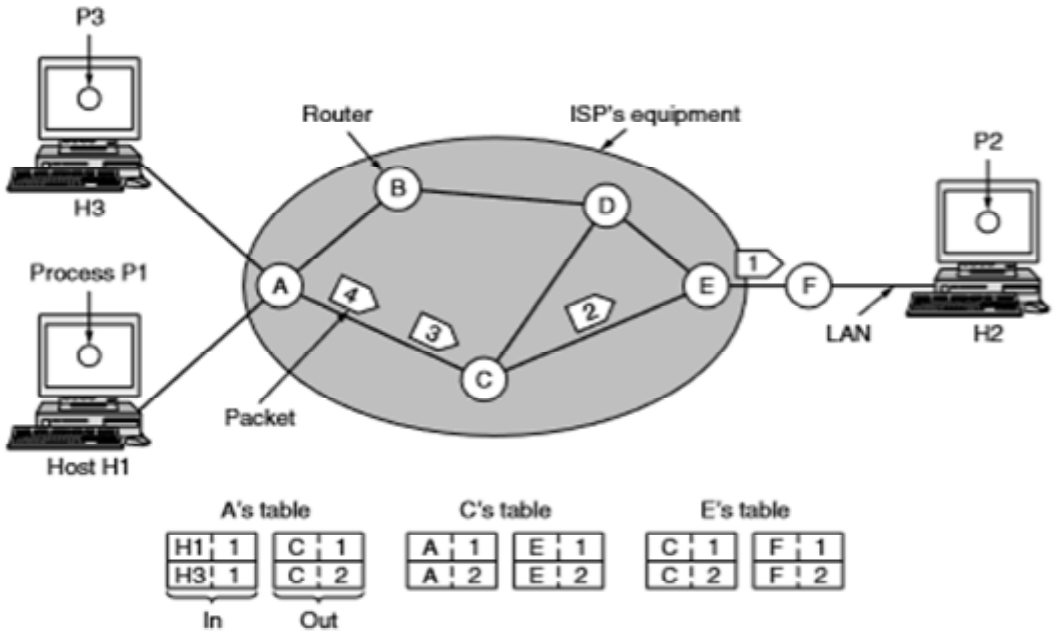


Figure 10.3 Routing within a virtual-circuit network

As an example, consider the situation shown in Fig. 10.3. Here, host H1 has established connection 1 with host H2. This connection is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packetbearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.

An example of a connection-oriented network service is MPLS (MultiProtocol Label Switching). It is used within ISP networks in the Internet, with IP packets wrapped in an MPLS header having a 20-bit connection identifier or label. MPLS is often hidden from customers, with the ISP establishing long-term connections for large amounts of traffic, but it is increasingly being used to help when quality of service is important but also with other ISP traffic management tasks.

10.3.5 Comparison of Virtual-Circuit and Datagram Networks

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize both sets of arguments. The major issues are listed in Fig. 10.4.

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Figure 10.4 Comparison of datagram and virtual-circuit networks

10.4 Routing Algorithms

The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network segment. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the already established route. The latter case is sometimes called session routing because a route remains in force for an entire session.

It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is forwarding. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Routing algorithms can be grouped into two major classes: nonadaptive and adaptive. Nonadaptive algorithms do not base their routing decisions on any measurements or estimates of the current topology and traffic. Instead, the choice of the route to use to get from I to J (for all I and J) is computed in advance, offline, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing. Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well. These dynamic routing algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., when the topology changes, or every ?T seconds as the load changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

In the following sections, we will discuss a variety of routing algorithms. The algorithms cover delivery models besides sending a packet from a source to a destination. Sometimes the goal is to send the packet to multiple, all, or one of a set of destinations. All of the routing algorithms make decisions based on the topologyand the routing algorithms defer the possibility of decisions based on the traffic levels.

10.4.1 The Optimality Principle

Before we get into specific algorithms, it may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle (Bellman,1957). It states that if router J is on the optimal path from router I to router K,then the optimal path from J to K also falls along the same

route. To see this, call the part of the route from I to Jr1 and the rest of the route r2 . If a route better than r2 existed from J to K, it could be concatenated with r1 to improve the routefrom I to K, contradicting our statement that r1 r 2 is optimal.

As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at thedestination. Such a tree is called a sink tree and is illustrated in Fig. 10-5(b),where the distance metric is the number of hops. The goal of all routing algorithmsis to discover and use the sink trees for all routers.

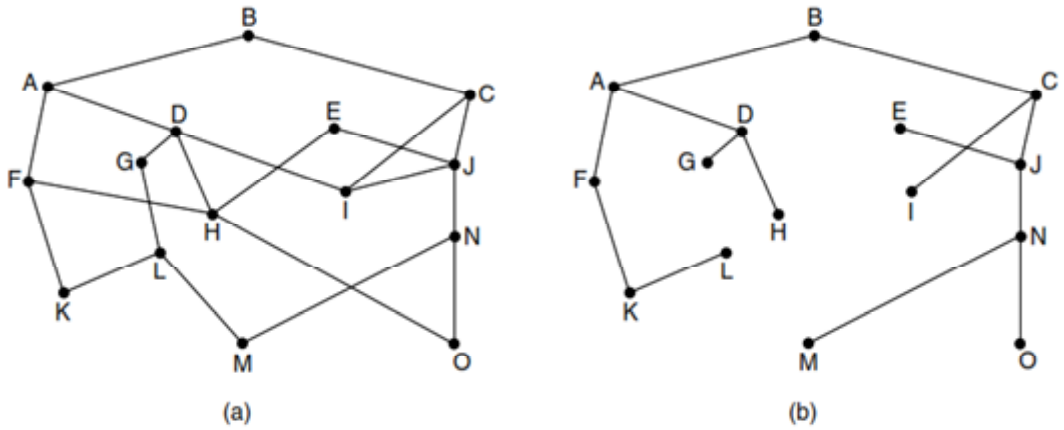


Figure 10.5 (a) A network. (b) A sink tree for router B

Note that a sink tree is not necessarily unique; other trees with the same pathlengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a DAG (Directed Acyclic Graph). DAGs have no loops.

10.4.2 Shortest Path Algorithm

Let us begin our study of routing algorithms with a simple technique for computing optimal paths given a complete picture of the network. These paths are the ones that we want a distributed routing algorithm to find, even though not all routers may know all of the details of the network.

The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

The concept of a shortest path deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths ABCand ABE in Fig. 10.6 are equally long. Another metric is the geographic distance kilometers, in which case ABC is clearly much longer than ABE (assuming thefigure is drawn to scale).

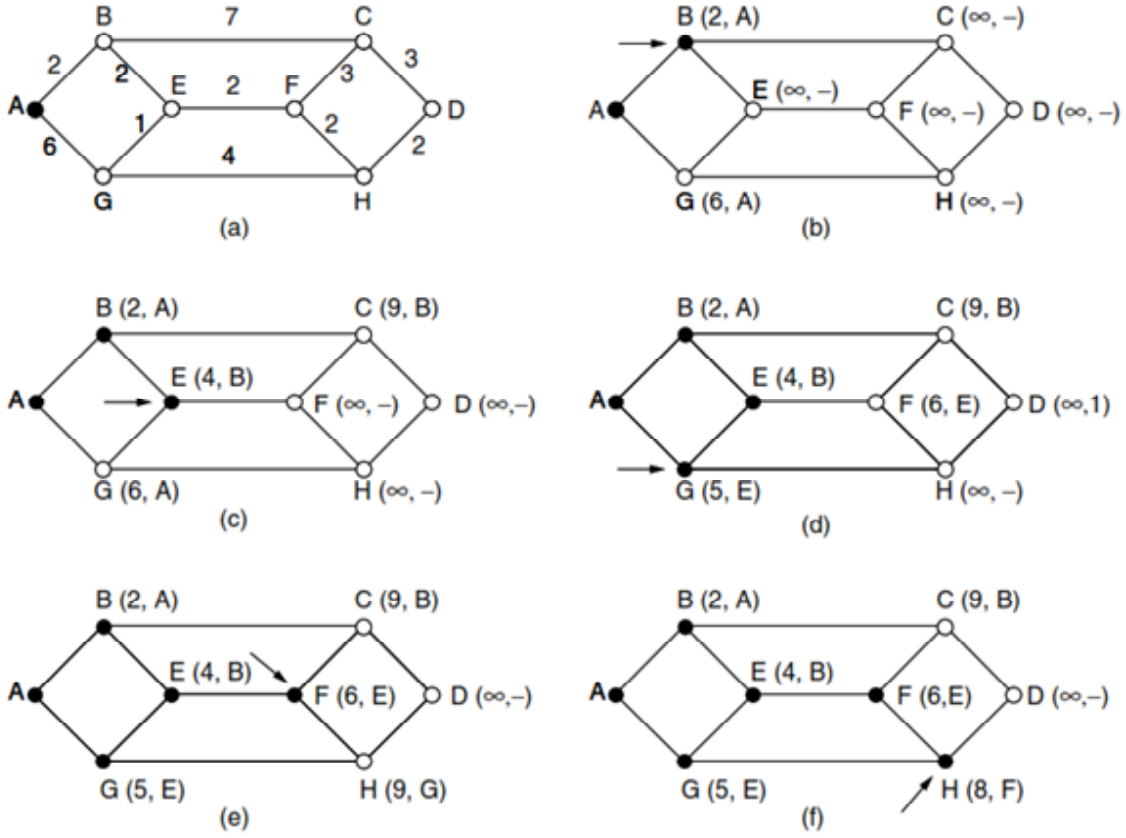


Figure 10.6 The first six steps used in computing the shortest path from A to D.

The arrows indicate the working node

However, many other metrics besides hops and physical distance are also possible. For example, each edge could be labeled with the mean delay of a standard packet, as measured by hourly runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest edges or kilometers.

In the general case, the labels on the edges could be computed as a function ofdistance, bandwidth, average traffic, communication cost, measured delay,other factors. By changing

the weighting function, the algorithm would then compute the "shortest" path measured according to any one of a number of criteriaor to a combination of criteria.

Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to Dijkstra (1959) and finds the shortest paths between a source and all destinations in the network. Each node is labeled (in parentheses) with its distance from the source node along the best known path.distances must be non-negative, as they will be if they are based on real quantities like bandwidth and delay. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent.Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent andnever changed thereafter.

To illustrate how the labeling algorithm works, look at the weighted,undirected graph of Fig. 10.6(a), where the weights represent, for example, distance.We want to find the shortest path from A to D. We start out by marking node A as permanent, indicated by a filled-in circle. Then we examine, in turn,each of the nodes adjacent to A (the working node), relabeling each one with the distance to A. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. If the network had more than one shortest path from A to D and we wanted to find all ofthem, we would need to remember all of the probe nodes that could reach a node with the same distance.

Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. 10.6(b). This one becomes the new working node.

We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled. After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure 10.6 shows the first six steps of the algorithm.

To see why the algorithm works, look at Fig. 10.6(c). At this point we havejust made E permanent. Suppose that there were a shorter path than ABE, say AXYZE (for some X and Y). There are two possibilities: either node Z has already been made permanent, or it has not been. If it has, then E has already been probed(on the round following the one when Z was made permanent), so the AXYZE path has not escaped our attention and thus cannot be a shorter path.

Now consider the case where Z is still tentatively labeled. If the label at Z is greater than or equal to that at E, then AXYZE cannot be a shorter path than ABE.If the label is less than that of E, then Z and not E will become permanent first, allowing E to be probed from Z.

10.4.3 Flooding

When a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network. A simple local technique is flooding, in which every incoming packet is sent out on every outgoing line except the one it arrived on.

Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet that is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is,can initialize the counter to the worst case, namely, the full diameter of the network.

Flooding with a hop count can produce an exponential number of duplicate packets as the hop count grows and routers duplicate packets they have seen before.A better technique for damming the flood is to have routers keep track of which packets have been flooded, to avoid sending them out a second time. One way to achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

To prevent the list from growing without bound, each list should be augmented by a counter, k, meaning that all sequence numbers through k have been seen. When a packet

comes in, it is easy to check if the packet has already been flooded (by comparing its sequence number to k; if so, it is discarded. Furthermore,the full list below k is not needed, since k effectively summarizes it.

Flooding is not practical for sending most packets, but it does have some important uses. First, it ensures that a packet is delivered to every node in the network.This may be wasteful if there is a single destination that needs the packet,but it is effective for broadcasting information. In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property.

Second, flooding is tremendously robust. Even if large numbers of routers are blown to bits (e.g., in a military network located in a war zone), flooding will finda path if one exists, to get a packet to its destination. Flooding also requires little in the way of setup. The routers only need to know their neighbors. This means that flooding can be used as a building block for other routing algorithms that are more efficient but need more in the way of setup. Flooding can also be used as a metric against which other routing algorithms can be compared. Flooding always chooses the shortest path because it chooses every possible path in parallel. Consequently,no other algorithm can produce a shorter delay.

10.4.4 Distance Vector Routing

Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology. Two dynamic algorithms in particular, distance vector routing and link state routing, are the most popular.

A distance vector routing algorithm operates by having each router maintaintable (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination.

The distance vector routing algorithm is sometimes called by other names,most commonly the distributed Bellman-Ford routing algorithm. In distance vector routing, each router maintains a routing table indexed by,and containing one entry for each router in the network. This entry

has two parts:preferred outgoing line to use for that destination and an estimate of the distanceto that destination. The distance might be measured as the number of hopsusing another metric.

The router is assumed to know the "distance" to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is propagation delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can.

10.4.5 Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed. Consequently, it was replaced by entirely new algorithm, now called link state routing. Variants of link stater outing called IS-IS and OSPF are the routing algorithms that are most widely used inside large networks and the Internet today.

The idea behind link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:

1. Discover its neighbors and learn their network addresses.
2. Set the distance or cost metric to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to and receive packets from all other routers.
5. Compute the shortest path to every other router.

In effect, the complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router.

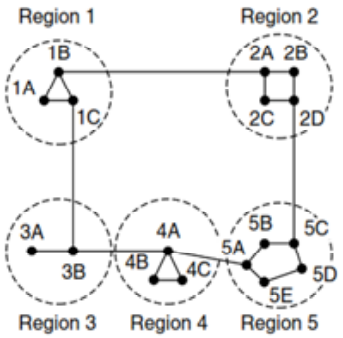
10.4.6 Hierarchical Routing

As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them. At a certain point, the network

may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the telephone network.

When hierarchical routing is used, the routers are divided into what we will call regions. Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions. When different networks are interconnected, it is natural to regard each one as a separate region to free the routers in one network from having to know the topological structure of the other ones.For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups, and so on, until we run out of names for aggregations.

Figure 10.7 gives a quantitative example of routing in a two-level hierarchy with five regions. The full routing table for router 1A has 17 entries, as shown inFig. 10.7(b). When routing is done hierarchically, as in Fig. 10.7(c), there are entries for all the local routers, as before, but all other regions are condensed into asingle router, so all traffic for region 2 goes via the 1B-2A line, but the rest of the remote traffic goes via the 1C-3B line. Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Figure 10.7 Hierarchical routing

When a single network becomes very large, an interesting question is "how many levels should the hierarchy have?" For example, consider a network with 720 routers. If there is no hierarchy, each router needs 720 routing table entries.If the network is partitioned into 24 regions of 30 routers each, each router needs 30 local entries plus 23 remote entries for a total of 53 entries. If a three-level hierarchy is chosen, with 8 clusters each containing 9 regions of 10 routers, each router needs 10 entries for local routers, 8 entries for routing to other regionswithin its own cluster, and 7 entries for distant clusters, for a total of 25 entries.Kamoun and Kleinrock (1979) discovered that the optimal number of levels for an N router network is ln N, requiring a total of e ln N entries per router. They havealso shown that the increase in effective mean path length caused by hierarchical routing is sufficiently small that it is usually acceptable.

10.4.7 Broadcast Routing

In some applications, hosts need to send messages to many or all other hosts.For example, a service distributing weather reports, stock market updates, or live radio programs might work best by sending to all machines and letting those thatare interested read the data. Sending a packet to all destinations simultaneously iscalled broadcasting. Various methods have been proposed for doing it.

One broadcasting method that requires no special features from the network is the source to simply send a distinct packet to each destination. Not only is the method wasteful of bandwidth and slow, but it also requires the source to have acomplete list of all destinations. This method is not desirable in practice, eventhough it is widely applicable.

An improvement is multidestination routing, in which each packet contains either a list of destinations or a bit map indicating the desired destinations. When packet arrives at a router, the router checks all the destinations to determine the of output lines that will be needed. The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line. In effect, the destination set is partitioned among the output lines. After a sufficient number of hops, each packet will carry only one destination like a normal packet. Multidestination routing is like using separately addressed packets, except that when several packets must follow the same route, one of them pays full fare and the rest ride free. The network bandwidth is therefore used more efficiently. However,

this scheme still requires the source to know all the destinations, plus it is as much work for a router to determine whereto send one multidestination packet as it is for multiple distinct packets.

10.4.8 Multicast Routing

Some applications, such as a multiplayer game or live video of a sports event streamed to many viewing locations, send packets to multiple receivers. Unlessthe group is very small, sending a distinct packet to each receiver is expensive.On the other hand, broadcasting a packet is wasteful if the group consists of, say,1000 machines on a million-node network, so that most receivers are not interested in the message. Thus, we need a way to send messages to well-defined groups that are numerically large in size but small compared to the network as a whole.

Sending a message to such a group is called multicasting, and the routing algorithm used is called multicast routing. All multicasting schemes require someway to create and destroy groups and to identify which routers are members of agroup. How these tasks are accomplished is not of concern to the routing algorithm.For now, we will assume that each group is identified by a multicast address and that routers know the groups to which they belong.

Multicast routing schemes build on the broadcast routing schemes we have ready studied, sending packets along spanning trees to deliver the packets to members of the group while making efficient use of bandwidth. However, best spanning tree to use depends on whether the group is dense, with receivers scattered over most of the network, or sparse, with much of the network not longing to the group. In this section we will consider both cases.

If the group is dense, broadcast is a good start because it efficiently gets packet to all parts of the network. But broadcast will reach some routers that not members of the group, which is wasteful. The solution explored by Deeringand Cheriton (1990) is to prune the broadcast spanning tree by removing links not lead to members. The result is an efficient multicast spanning tree.

10.4.9 Anycast Routing

So far, we have covered delivery models in which a source sends to a single destination (called unicast), to all destinations (called broadcast), and to a group destinations (called

multicast). Another delivery model, called anycast is sometimes also useful. In anycast, a packet is delivered to the nearest member ofgroup (Partridge et al., 1993). Schemes that find these paths are called anycast routing.

Sometimes nodes provide a service, such as time of day or content distribution for which it is getting the right information all that matters, not the node that is contacted; any node will do. For example, anycast is used in the Internet as part of DNS. Luckily, we will not have to devise new routing schemes for anycast because regular distance vector and link state routing can produce anycast routes. Suppose we want to anycast to the members of group 1. They will all be given the address"1," instead of different addresses. Distance vector routing will distribute vectors as usual, and nodes will choose the shortest path to destination 1. This will result in nodes sending to the nearest instance of destination 1. This procedure works because the routing protocol does not realize that there are multiple instances of destination 1. That is, it believes that all theinstances of node 1 are the same node.

10.5 Summary

The network layer provides services to the transport layer. It can be based on either datagrams or virtual circuits. In both cases, its main job is routing packets from the source to the destination. In datagram networks, a routing decision is made on every packet. In virtual-circuit networks, it is made when the virtual circuitis set up.

Many routing algorithms are used in computer networks. Flooding is a simple algorithm to send a packet along all paths. Most algorithms find the shortest path and adapt to changes in the network topology. The main algorithms are distance vector routing and link state routing. Most actual networks use one of these.Other important routing topics are the use of hierarchy in large networks, routing for mobile hosts, and broadcast, multicast, and anycast routing.

10.6 Model Questions

1. Explain network layer design issues.
2. Describe the Distance vector routing algorithm with an example subnet.
3. Explain the following routing algorithms:

(a) Hierarchical routing

(b) Link state routing

4. Write notes on shortest path routing.

5. Explain about broadcast routing and anycast routing techniques.

LESSON-11
CONGESTION CONTROL ALGORITHMS

Structure

11.1 Introduction

Too many packets present in the network causes packet delay and loss that degrades performance. This situation is called congestion. The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together. In this lesson, we will look at the network aspects of congestion.

11.2 Objectives

- To explain about the congestion control algorithms.
- To discuss the Internet Protocol and it's versions.
- To illustrate the various types of Internet Control Protocols.

11.3 Approaches to Congestion Control

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load. As shown in Fig. 11.1, these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.

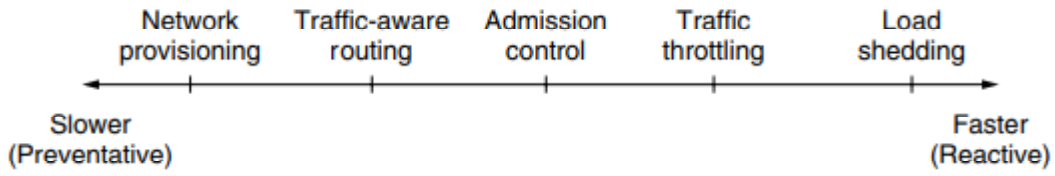


Figure 11.1 Timescales of approaches to congestion control

The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources can be added dynamically when there is serious congestion, for example, turning on spare routers or enabling lines that are normally used only as backups (to make the system fault tolerant) or purchasing bandwidth on the open market. More often, links and routers that are regularly heavily utilized are upgraded at the earliest opportunity. This is called provisioning and happens on a time scale of months, driven by long-term traffic trends.

To make the most of the existing network capacity, routes can be tailored to traffic patterns that change during the day as network users wake and sleep in different time zones. For example, routes may be changed to shift traffic away from heavily used paths by changing the shortest path weights. Some local radio stations have helicopters flying around their cities to report on

road congestion to make it possible for their mobile listeners to route their packets (cars) around hotspots. This is called traffic-aware routing. Splitting traffic across multiple paths is also helpful.

However, sometimes it is not possible to increase capacity. The only way then to beat back the congestion is to decrease the load. In a virtual-circuit network, new connections can be refused if they would cause the network to become congested. This is called admission control.

At a finer granularity, when congestion is imminent the network can deliver feedback to the sources whose traffic flows are responsible for the problem. The network can request these sources to throttle their traffic, or it can slow down the traffic itself. Two difficulties with this approach are how to identify the onset of congestion, and how to inform the source that needs to slow down. To tackle the first issue, routers can monitor the average load, queueing delay, or packet loss. In all cases, rising numbers indicate growing congestion. To tackle the second issue, routers must participate in a feedback loop with the sources. For a scheme to work correctly, the time scale must be adjusted carefully. If every time two packets arrive in a row, a router yells STOP and every time a router is idle for 20 μsec, it yells GO, the system will oscillate wildly and never converge. On the other hand, if it waits 30 minutes to make sure before saying anything, the congestion-control mechanism will react too sluggishly to be of any use. Delivering timely feedback is a nontrivial matter. An added concern is having routers send more messages when the network is already congested.

Finally, when all else fails, the network is forced to discard packets that it cannot deliver. The general name for this is load shedding. A good policy for choosing which packets to discard can help to prevent congestion collapse.

11.4 Traffic-Aware Routing

The first approach we will examine is traffic-aware routing. The routing schemes used fixed link weights. These schemes adapted to changes in topology, but not to changes in load. The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion.

The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal.

Traffic-aware routing was used in the early Internet. Consider the network of Fig. 11.2, which is divided into two parts, East and West, connected by two links, CF and EI. Suppose that most of the traffic between East and West is using link CF, and, as a result, this link is heavily loaded with long delays including queueing delay in the weight used for the shortest path calculation will make EI more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over EI, loading this link. Consequently, in the next update, CF will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.

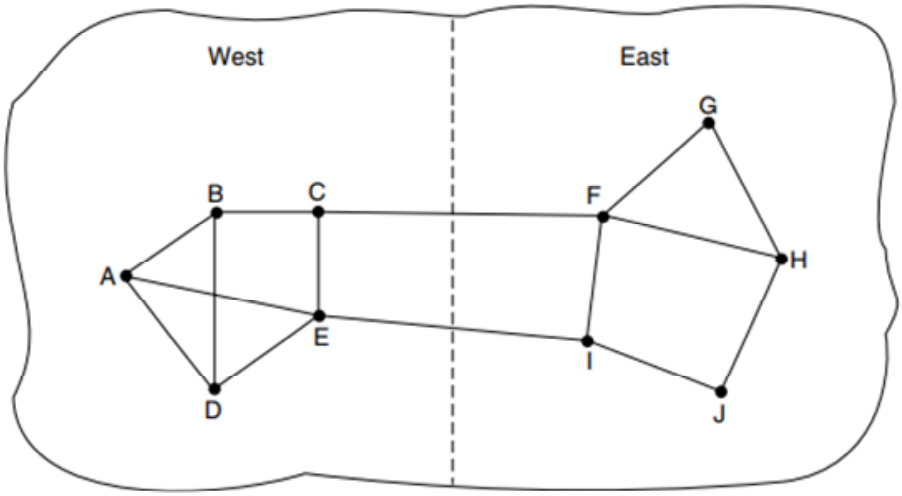


Figure 11.2 A network in which the East and West parts are connected by two links

If load is ignored and only bandwidth and propagation delay are considered, this problem does not occur. Attempts to include load but change weights within a narrow range only slow down routing oscillations. Two techniques can contribute to a successful solution. The first is multipath routing, in which there can be multiple paths from a source to a destination. In our example this means that the traffic can be spread across both of the East to West links. The second one is for the routing scheme to shift traffic across routes slowly enough that it is able to converge.

Given these difficulties, in the Internet routing protocols do not generally adjust their routes depending on the load. Instead, adjustments are made outside the routing protocol by slowly changing its inputs. This is called traffic engineering.

11.5 Admission Control

One technique that is widely used in virtual-circuit networks to keep congestion at bay is admission control. The idea is simple: do not set up a new virtual circuit unless the network can carry the added traffic without becoming congested. Thus, attempts to set up a virtual circuit may fail. This is better than the alternative, as letting more people in when the network is busy just makes matters worse. By analogy, in the telephone system, when a switch gets overloaded it practices admission control by not giving dial tones.

The trick with this approach is working out when a new virtual circuit will lead to congestion. The task is straightforward in the telephone network because of the fixed bandwidth of calls (64 kbps for uncompressed audio). However, virtual circuits in computer networks come in all shapes and sizes. Thus, the circuit must come with some characterization of its traffic if we are to apply admission control.

Traffic is often described in terms of its rate and shape. The problem of how to describe it in a simple yet meaningful way is difficult because traffic is typically bursty-the average rate is only half the story. For example, traffic that varies while browsing the Web is more difficult to handle than a streaming movie with the same long-term throughput because the bursts of Web traffic are more likely to congest routers in the network. A commonly used descriptor that captures this effect is the leaky bucket or token bucket. A leaky bucket has two parameters that bound the average rate and the instantaneous burst size of traffic.

11.6 Traffic Throttling

In the Internet and many other computer networks, senders adjust their transmissions to send as much traffic as the network can readily deliver. In this setting, the network aims to operate just before the onset of congestion. When congestion is imminent, it must tell the senders to throttle back their transmissions and slow down. This feedback is business as usual

rather than an exceptional situation. The term congestion avoidance is sometimes used to contrast this operating point with the one in which the network has become (overly) congested.

Let us now look at some approaches to throttling traffic that can be used in both datagram networks and virtual-circuit networks. Each approach must solve two problems. First, routers must determine when congestion is approaching, ideally before it has arrived. To do so, each router can continuously monitor the resources it is using. Three possibilities are the utilization of the output links, the buffering of queued packets inside the router, and the number of packets that are lost due to insufficient buffering. Of these possibilities, the second one is the most useful. Averages of utilization do not directly account for the burstiness of most traffic-a utilization of 50% may be low for smooth traffic and too high for highly variable traffic. Counts of packet losses come too late. Congestion has already set in by the time that packets are lost.

The second problem is that routers must deliver timely feedback to the senders that are causing the congestion. Congestion is experienced in the network, but relieving congestion requires action on behalf of the senders that are using the network. To deliver feedback, the router must identify the appropriate senders. It must then warn them carefully, without sending many more packets into the already congested network. Different schemes use different feedback mechanisms, as we will now describe.

11.6.1 Choke Packets

The most direct way to notify a sender of congestion is to tell it directly. In this approach, the router selects a congested packet and sends a choke packet back to the source host, giving it the destination found in the packet. The original packet may be tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and then forwarded in the usual way. To avoid increasing load on the network during a time of congestion, the router may only send choke packets at a low rate.

When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination, for example, by 50%. In a datagram network, simply picking packets at random when there is congestion is likely to cause choke packets to be sent to fast senders, because they will have the most packets in the queue. The feedback implicit in this protocol can

help prevent congestion yet not throttle any sender unless it causes trouble. For the same reason, it is likely that multiple choke packets will be sent to a given host and destination. The host should ignore these additional chokes for the fixed time interval until its reduction in traffic takes effect. After that period, further choke packets indicate that the network is still congested.

11.6.2 Explicit Congestion Notification

Instead of generating additional packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion. When the network delivers the packet, the destination can note that there is congestion and inform the sender when it sends a reply packet. The sender can then throttle its transmissions as before.This design is called ECN (Explicit Congestion Notification) and is used in the Internet. It is a refinement of early congestion signaling protocols.

11.7 Load Shedding

When none of the above methods make the congestion disappear, routers can bring out the heavy artillery: load shedding. Load shedding is a fancy way of saying that when routers are being inundated by packets that they cannot handle, they just throw them away. The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.

The key question for a router drowning in packets is which packets to drop. The preferred choice may depend on the type of applications that use the network. For a file transfer, an old packet is worth more than a new one. This is because dropping packet 6 and keeping packets 7 through 10, for example, will only force the receiver to do more work to buffer data that it cannot yet use. In contrast, for real-time media, a new packet is worth more than an old one. This is because packets become useless if they are delayed and miss the time at which they must be played out to the user.

11.8 The Network Layer in the Internet

It is now time to discuss the network layer of the Internet in detail. But before getting into specifics, it is worth taking a look at the principles that drove its design in the past and made it

the success that it is today. We will now summarize what we consider to be the top 10 principles (from most important to least important).

1. Make sure it works. Do not finalize the design or standard until multiple prototypes have successfully communicated with each other. All too often, designers first write a 1000-page standard, get it approved, then discover it is deeply flawed and does not work. Then they write version 1.1 of the standard. This is not the way to go.
2. Keep it simple. When in doubt, use the simplest solution. William of Occam stated this principle (Occam's razor) in the 14th century. Put in modern terms: fight features. If a feature is not absolutely essential, leave it out, especially if the same effect can be achieved by combining other features.
3. Make clear choices. If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best. Designers should strongly resist this tendency. Just say no.
4. Exploit modularity. This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances require one module or layer to be changed, the other ones will not be affected.
5. Expect heterogeneity. Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.
6. Avoid static options and parameters. If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value rather than defining fixed choices.
7. Look for a good design;it need not be perfect. Often, the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.
8. Be strict when sending and tolerant when receiving. In other words, send only packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.

9. Think about scalability. If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.
10. Consider performance and cost. If a network has poor performance or outrageous costs, nobody will use it.

The glue that holds the whole Internet together is the network layer protocol, IP (Internet Protocol). Unlike most older network layer protocols, IP was designed from the beginning with internetworking in mind. A good way to think of the network layer is this: its job is to provide a best-effort way to transport packets from source to destination, without regard to whether these machines are on the same network or whether there are other networks in between them.

Communication in the Internet works as follows. The transport layer takes data streams and breaks them up so that they may be sent as IP packets. In theory, packets can be up to 64 KB each, but in practice they are usually not more than 1500 bytes (so they fit in one Ethernet frame). IP routers forward each packet through the Internet, along a path from one router to the next, until the destination is reached. At the destination, the network layer hands the data to the transport layer, which gives it to the receiving process. When all the pieces finally get to the destination machine, they are reassembled by the network layer into the original datagram. This datagram is then handed to the transport layer. It is the job of the IP routing protocols to decide which paths to use.

11.8.1 IP Versions

There are two versions of IP in use today, IPv4 and IPv6. The original IPv4 protocol is still used today on both the internet, and many corporate networks. However, the IPv4 protocol only allowed for 232 addresses. This, coupled with how addresses were allocated, led to a situation where there would not be enough unique addresses for all devices connected to the internet.

IPv6 was developed by the Internet Engineering Task Force (IETF), and was formalized in 1998. This upgrade substantially increased the available address space and allowed for 2128 addresses. In addition, there were changes to improve the efficiency of IP packet headers, as well as improvements to routing and security.

11.8.2 IP Addresses

An IP address (internet protocol address) is a numerical representation that uniquely identifies a specific interface on the network.IP addresses are binary numbers but are typically expressed in decimal form (IPv4) or hexadecimal form (IPv6) to make reading and using them easier for humans.IP stands for Internet Protocol and describes a set of standards and requirements for creating and transmitting data packets, or datagrams, across networks. The Internet Protocol (IP) is part of the Internet layer of the Internet protocol suite. In the OSI model, IP would be considered part of the network layer. IP is traditionally used in conjunction with a higher-level protocol, most notably TCP. The IP standard is governed by RFC 791.

IP is a connectionless protocol that is datagram-oriented., so each packet must contain the source IP address, destination IP address, and other data in the header to be successfully delivered.Combined, these factors make IP an unreliable, best effort delivery protocol. Error correction is handled by upper level protocols instead. These protocols include TCP, which is a connection-oriented protocol, and UDP, which is a connectionless protocol.Most internet traffic is TCP/IP.

IPv4 addresses

IPv4 addresses are actually 32-bit binary numbers, consisting of the two subaddresses (identifiers) mentioned above which, respectively, identify the network and the host to the network, with an imaginary boundary separating the two. An IP address is, as such, generally shown as 4 octets of numbers from 0-255 represented in decimal form instead of binary form.For example, the address 168.212.226.204 represents the 32-bit binary number 10101000.11010100.11100010.11001100.The binary number is important because that will determine which class of network the IP address belongs to.

An IPv4 address is typically expressed in dotted-decimal notation, with every eight bits (octet) represented by a number from one to 255, each separated by a dot. An example IPv4 address would look like this: 192.168.17.43. IPv4 addresses are composed of two parts. The first numbers in the address specify the network, while the latter numbers specify the specific host. A subnet mask specifies which part of an address is the network part, and which part addresses the specific host.

A packet with a destination address that is not on the same network as the source address will be forwarded, or routed, to the appropriate network. Once on the correct network, the host part of the address determines which interface the packet gets delivered to.

Subnet masks

A single IP address identifies both a network, and a unique interface on that network. A subnet mask can also be written in dotted decimal notation and determines where the network part of an IP address ends, and the host portion of the address begins.

When expressed in binary, any bit set to one means the corresponding bit in the IP address is part of the network address. All the bits set to zero mark the corresponding bits in the IP address as part of the host address.The bits marking the subnet mask must be consecutive ones. Most subnet masks start with 255 and continue on until the network mask ends. A Class C subnet mask would be 255.255.255.0.

IP Address Classes

Before variable length subnet masks allowed networks of any size to be configured, the IPv4 address space was broken into five classes.

Class A

In a Class A network, the first eight bits, or the first dotted decimal, is the network part of the address, with the remaining part of the address being the host part of the address. There are 128 possible Class A networks. The range of this type of networks is from 0.0.0.0 to 127.0.0.0. Example for a Class A IP address: 2.134.213.2

Class B

In a Class B network, the first 16 bits are the network part of the address. All Class B networks have their first bit set to 1 and the second bit set to 0. In dotted decimal notation, that makes 128.0.0.0 to 191.255.0.0 as Class B networks. There are 16,384 possible Class B networks.Example for a Class B IP address: 135.58.24.17

Class C

In a Class C network, the first two bits are set to 1, and the third bit is set to 0. That makes the first 24 bits of the address the network address and the remainder as the host address. Class C network addresses range from 192.0.0.0 to 223.255.255.0. There are over 2 million possible Class C networks.Example for a Class C IP address: 192.168.178.1

Class D

Class D addresses are used for multicasting applications. Unlike the previous classes, the Class D is not used for "normal" networking operations. Class D addresses have their first three bits set to "1" and their fourth bit set to "0". Class D addresses are 32-bit network addresses, meaning that all the values within the range of 224.0.0.0 - 239.255.255.255 are used to uniquely identify multicast groups. There are no host addresses within the Class D address space, since all the hosts within a group share the group's IP address for receiver purposes.Example for a Class D IP address:227.21.6.173

Class E

Class E networks are defined by having the first four network address bits as 1. That encompasses addresses from 240.0.0.0 to 255.255.255.255. While this class is reserved, its usage was never defined. As a result, most network implementations discard these addresses as illegal or undefined. The exception is 255.255.255.255, which is used as a broadcast address.Example for a Class D IP address:243.164.89.28

Private addresses

Within the address space, certain networks are reserved for private networks. Packets from these networks are not routed across the public internet. This provides a way for private networks to use internal IP addresses without interfering with other networks. The private networks are: 10.0.0.1 - 10.255.255.255,

172.16.0.0-172.32.255.255 and 192.168.0.0 - 192.168.255.255.

Special addresses

Certain IPv4 addresses are set aside for specific uses:

127.0.0.0	Loopback address (the host's own interface)
224.0.0.0	IP Multicast
255.255.255.255	Broadcast (sent to all interfaces on network)

IPv6 addresses

To avoid the seemingly reoccurring issue in technology, where a specification's limitation seems more than sufficient at the time, but inevitably becomes too small, the designers of IPv6 created an enormous address space for IPv6. The address size was increased from 32 bits in IPv4 to 128 bits in IPv6.The IPv6 has a theoretical limit of 3.4 x 1038 addresses. IPv6 addresses are represented by eight sets of four hexadecimal digits, and each set of numbers is separated by a colon. An example IPv6 address would look like this:

2DAB:FFFF:0000:3EAE:01AA:00FF:DD72:2C4A

11.9 Internet Control Protocol

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP, ARP, and DHCP. In this section, we will look at each of these in turn, describing the versions that correspond to IPv4 because they are the protocols that are in common use. ICMP and DHCP have similar versions for IPv6; the equivalent of ARP is called NDP (Neighbor Discovery Protocol) for IPv6.

11.7.1 IMCP-The Internet Control Message Protocol

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the ICMP (Internet Control Message Protocol). ICMP is also used to test the Internet. About a dozen types of ICMP messages are defined. Each ICMP message type is carried encapsulated in an IP packet. The most important ones are listed in Fig. 11.3.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

Figure 11.3 The principal ICMP message types

The DESTINATION UNREACHABLE message is used when the router cannot locate the destination or when a packet with the DF bit cannot be delivered because a "small-packet" network stands in the way.

The TIME EXCEEDED message is sent when a packet is dropped because its TtL (Time to live) counter has reached zero. This event is a symptom that packets are looping, or that the counter values are being set too low.

The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or possibly in the software of a router transited.

The SOURCE QUENCH message was long ago used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down.

The REDIRECT message is used when a router notices that a packet seems to be routed incorrectly. It is used by the router to tell the sending host to update to a better route.

The ECHO and ECHO REPLY messages are sent by hosts to see if a given destination is reachable and currently alive. Upon receiving the ECHO message, the destination is expected to send back an ECHO REPLY message. These messages are used in the ping utility that checks if a host is up and on the Internet.

The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility can be used to measure network performance.

The ROUTER ADVERTISEMENT and ROUTER SOLICITATION messages are used to let hosts find nearby routers. A host needs to learn the IP address of at least one router to be able to send packets off the local network. In addition to these messages, others have been defined.

11.7.2 ARP-The Address Resolution Protocol

Although every machine on the Internet has one or more IP addresses, these addresses are not sufficient for sending packets. Data link layer NICs (Network Interface Cards) such as Ethernet cards do not understand Internet addresses. In the case of Ethernet, every NIC ever manufactured comes equipped with a unique 48-bit Ethernet address. Manufacturers of Ethernet NICs request a block of Ethernet addresses from IEEE to ensure that no two NICs have the same address. The NICs send and receive frames based on 48-bit Ethernet addresses. They know nothing at all about 32-bit IP addresses.

The Address Resolution Protocol (ARP) is a protocol used by the Internet Protocol (IP), specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

The term address resolution refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

An Ethernet network uses two hardware addresses which identify the source and destination of each frame sent by the Ethernet. The destination address (all 1's) may also

identify a broadcast packet (to be sent to all connected computers). The hardware address is also known as the Medium Access Control (MAC) address, in reference to the standards which define Ethernet. Each computer's Network Interface Card (NIC)is allocated a globally unique 6 byte link address when the factory manufactures the card (stored in a PROM). This is the normal link source address used by an interface. A computer sends all packets which it creates with its own hardware source link address, and receives all packets which match the same hardware address in the destination field or one (or more) pre-selected broadcast/multicast addresses.

The Ethernet address is a link layer address and is dependent on the interface card which is used. IP operates at the network layer and is not concerned with the link addresses of individual nodes which are to be used.The ARP is therefore used to translate between the two types of address. The ARP client and server processes operate on all computers using IP over Ethernet. The processes are normally implemented as part of the software driver that drives the NIC.

There are four types of ARP messages that may be sent by the ARP protocol. These are identified by four values in the "operation" field of an ARP message. The types of message are:(1) ARP request (2) ARP reply (3) RARP request and (4) RARP reply.

11.7.3 DHCP-The Dynamic Host Configuration Protocol

ARP (as well as other Internet protocols) makes the assumption that hosts are configured with some basic information, such as their own IP addresses. How do hosts get this information? It is possible to manually configure each computer, but that is tedious and error-prone. There is a better way, and it is called DHCP (Dynamic Host Configuration Protocol).

With DHCP, every network must have a DHCP server that is responsible for configuration. When a computer is started, it has a built-in Ethernet or other link layer address embedded in the NIC, but no IP address. Much like ARP, the computer broadcasts a request for an IP address on its network. It does this by using a DHCP DISCOVER packet. This packet must reach the DHCP server. If that server is not directly attached to the network, the router will be configured to receive DHCP broadcasts and relay them to the DHCP server, wherever it is located.

When the server receives the request, it allocates a free IP address and sends it to the host in a DHCP OFFER packet (which again may be relayed via the router). To be able to do this work even when hosts do not have IP addresses, the server identifies a host using its Ethernet address.

An issue that arises with automatic assignment of IP addresses from a pool is for how long an IP address should be allocated. If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost. After a period of time, many addresses may be lost. To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called leasing. Just before the lease expires, the host must ask for a DHCP renewal. If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.

DHCP is described in RFCs 2131 and 2132. It is widely used in the Internet to configure all sorts of parameters in addition to providing hosts with IP addresses. As well as in business and home networks, DHCP is used by ISPs (Internet Service Provider) to set the parameters of devices over the Internet access link, so that customers do not need to phone their ISPs to get this information. Common examples of the information that is configured include the network mask, the IP address of the default gateway, and the IP addresses of DNS (Domain Name System)and time servers. DHCP has largely replaced earlier protocols (called RARP and BOOTP) with more limited functionality.

11.10 Summary

Networks can easily become congested, leading to increased delay and lost packets. Network designers attempt to avoid congestion by designing the network to have enough capacity, choosing uncongested routes, refusing to accept more traffic, signaling sources to slow down, and shedding load.

The Internet has a rich variety of protocols related to the network layer. These include the datagram protocol, IP, and associated control protocols such as ICMP, ARP, and DHCP.

11.11 Model Questions

1. Give a note on Congestion Prevention Policies.
2. Explain the procedure for preallocation of buffer and packet discarding.
3. Write notes on : (a) Congestion prevention policies. (b) Congestion control in virtual circuit subnets.
4. Describe the concept of packet discarding and choke packets.
5. Illustrate about Internet Control Protocols
6. Explain in detail about the two versions of Internet Protocol (IP).

LESSON-12
THE TRANSPORT LAYER

Structure

12.1 Introduction

Together with the network layer, the transport layer is the heart of the protocol hierarchy. The network layer provides end-to-end packet delivery using datagrams or virtual circuits. The transport layer builds on the network layer to provide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use. It provides the abstractions that applications need to use the network. Without the transport layer, the whole concept of layered protocols would make little sense. In this lesson, we will study the transport layer in detail, including its services and choice of API design to tackle issues of reliability, connections and congestion control, protocols such as TCP and UDP, and performance.

12.2 Objectives

- To discuss about the services provided by the transport layer.
- To explain the elements of Transport protocol.

12.3 The Transport Service

In the following sections, we will provide an introduction to the transport service. We look at what kind of service is provided to the application layer. To make the issue of transport

service more concrete, we will examine two sets of transport layer primitives. First comes a simple one to show the basic ideas. Then comes the interface commonly used in the Internet.

12.3.1 Services Provided tothe Upper Layers

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the transport entity. The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card. The first two options are most common on the Internet. The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 12.1.

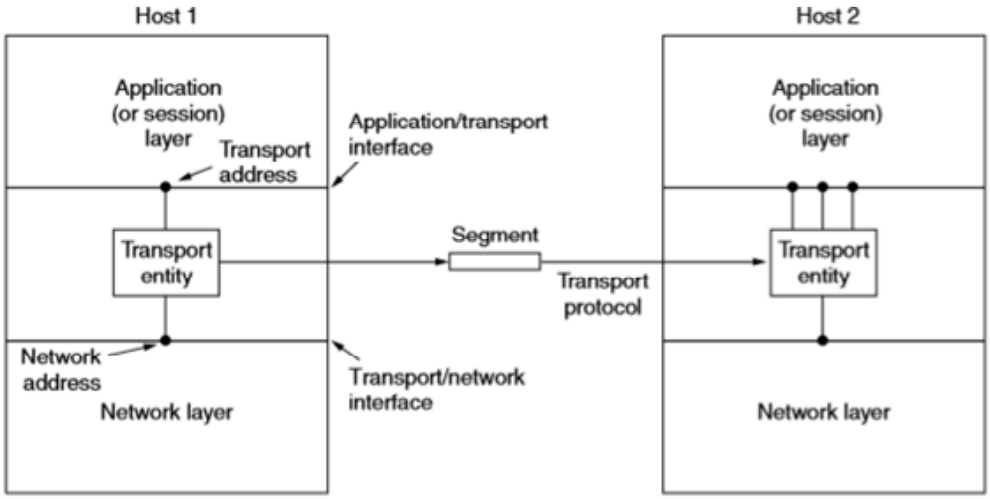


Figure 12.1 The network, transport, and application layers

Just as there are two types of network service, connection-oriented and connectionless, there are also two types of transport service. The connection-oriented transport service is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers. Furthermore, the connectionless transport service is also very similar to the connectionless network service. However, note that it can be difficult to provide a

connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear it down immediately afterwards.

In essence, the existence of the transport layer makes it possible for the transport service to be more reliable than the underlying network. Furthermore, the transport primitives can be implemented as calls to library procedures to make them independent of the network primitives. The network service calls may vary considerably from one network to another. Hiding the network service behind a set of transport service primitives ensures that changing the network merely requires replacing one set of library procedures with another one that does the same thing with a different underlying service.

There is a qualitative distinction between layers 1 through 4 on the one hand and layer(s) above 4 on the other. The bottom four layers can be seen as the transport service provider, whereas the upper layer(s) are the transport service user. This distinction of provider versus user has a considerable impact on the design of the layers and puts the transport layer in a key position, since it forms the major boundary between the provider and user of the reliable data transmission service. It is the level that applications see.

12.3.2 Transport Service Primitives

To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface. In this section, we will first examine a simple transport service and its interface to see the bare essentials.

The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks, warts and all. Real networks can lose packets, so the network service is generally unreliable.

The connection-oriented transport service, in contrast, is reliable. Of course, real networks are not error-free, but that is precisely the purpose of the transport layer-to provide a reliable service on top of an unreliable network.

A second difference between the network service and transport service is whom the services are intended for. The network service is used only by the transport entities. Few users write their own transport entities, and thus few users or programs ever see the bare network service. In contrast, many programs see the transport primitives. Consequently, the transport service must be convenient and easy to use.

To get an idea of what a transport service might be like, consider the five primitives listed in Fig. 12.2. This transport interface is truly bare bones, but it gives the essential flavor of what a connection-oriented transport interface has to do. It allows application programs to establish, use, and then release connections, which is sufficient for many applications.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

Figure 12.2 The primitives for a simple transport service

Thus, segments (exchanged by the transport layer) are contained in packets (exchanged by the network layer). In turn, these packets are contained in frames (exchanged by the data link layer). When a frame arrives, the data link layer processes the frame header and, if the destination address matches for local delivery, passes the contents of the frame payload field up to the network entity. The network entity similarly processes the packet header and then passes the contents of the packet payload up to the transport entity. This nesting is illustrated in Fig. 12.3.

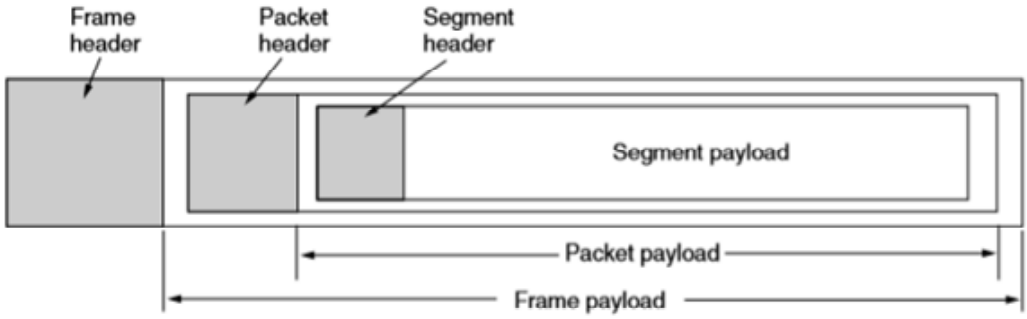


Figure 12.3 Nesting of segments, packets, and frames

Getting back to our client-server example, the client's CONNECT call causes a CONNECTION REQUEST segment to be sent to the server. When it arrives, thetransport entity checks to see that the server is blocked on a LISTEN (i.e., is interested in handling requests). If so, it then unblocks the server and sends a CONNECTION ACCEPTED segment back to the client. When this segment arrives, the client is unblocked and the connection is established.

Data can now be exchanged using the SEND and RECEIVE primitives. In the simplest form, either party can do a (blocking) RECEIVE to wait for the other party to do a SEND. When the segment arrives, the receiver is unblocked. It can then process the segment and send a reply. As long as both sides can keep track of whose turn it is to send, this scheme works fine.

Note that in the transport layer, even a simple unidirectional data exchange is more complicated than at the network layer. Every data packet sent will also be acknowledged (eventually). The packets bearing control segments are also acknowledged, implicitly or explicitly. These acknowledgements are managed by the transport entities, using the network layer protocol, and are not visible to the transport users. Similarly, the transport entities need to worry about timers and retransmissions. None of this machinery is visible to the transport users. To the transport users, a connection is a reliable bit pipe: one user stuffs bits in and they magically appear in the same order at the other end. This ability to hide complexity is the reason that layered protocols are such a powerful tool.

When a connection is no longer needed, it must be released to free up table space within the two transport entities. Disconnection has two variants: asymmetric and symmetric. In the asymmetric variant, either transport user can issue a DISCONNECT primitive, which results in a DISCONNECT segment being sent to the remote transport entity. Upon its arrival, the connection is released.

In the symmetric variant, each direction is closed separately, independently of the other one. When one side does a DISCONNECT, that means it has no more data to send but it is still willing to accept data from its partner. In this model, a connection is released when both sides have done a DISCONNECT.

12.3.3 Berkeley Sockets

Let us now briefly inspect another set of transport primitives, the socket primitives as they are used for TCP. Sockets were first released as part of the Berkeley UNIX 4.2BSD software distribution in 1983. They quickly became popular. The primitives are now widely used for Internet programming on many operating systems, especially UNIX-based systems, and there is a socket-style API for Windows called "winsock." The primitives are listed in Fig. 12.4.

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 12.4 The socket primitives for TCP

12.4 Elements of Transport Protocol

The transport service is implemented by a transport protocol used between the two transport entities. In some ways, transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues.

However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 12.5. At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. This difference has many important implications for the protocols.

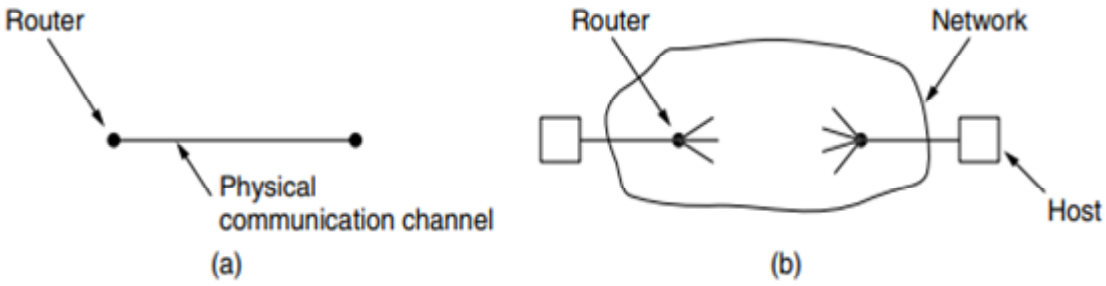


Figure 12.5 (a) Environment of the data link layer
(b) Environment of the transport layer

For one thing, over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to-each outgoing line leads directly to a particular router. In the transport layer, explicit addressing of destinations is required.

For another thing, the process of establishing a connection over the wire of Fig. 12.5(a) is simple: the other end is always there. Either way, there is not much to do. Even on wireless links, the process is not much different. Just sending a message is sufficient to have it reach all other destinations. If the message is not acknowledged due to an error, it can be resent. In the transport layer, initial connection establishment is complicated.

Another difference between the data link layer and the transport layer is the potential existence of storage capacity in the network. When a router sends a packet over a link, it may arrive or be lost, but it cannot bounce around for a while, go into hiding in a far corner of the world, and suddenly emerge after other packets that were sent much later. If the network uses datagrams, which are independently routed inside, there is a nonnegligible probability that a packet may take the scenic route and arrive late and out of the expected order, or even that duplicates of the packet will arrive. The consequences of the network's ability to delay and duplicate packets can sometimes be disastrous and can require the use of special protocols to correctly transport information.

A final difference between the data link and transport layers is one of degree rather than of kind. Buffering and flow control are needed in both layers, but the presence in the transport

layer of a large and varying number of connections with bandwidth that fluctuates as the connections compete with each other may require a different approach than we used in the data link layer. Some of the protocols discussed in data link layer allocate a fixed number of buffers to each line, so that when a frame arrives a buffer is always available. In the transport layer, the larger number of connections that must be managed and variations in the bandwidth each connection may receive make the idea of dedicating many buffers to each one less attractive.

12.4.1 Addressing

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called ports. We will use the generic term TSAP (Transport Service Access Point) to mean a specific endpoint in the transport layer. The analogous endpoints in the network layer (i.e., network layer addresses) are not-surprisingly called NSAPs (Network Service Access Points). IP addresses are examples of NSAPs.

Figure 12.6 illustrates the relationship between the NSAPs, the TSAPs, and a transport connection. Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host, as shown. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport endpoints that share that NSAP.

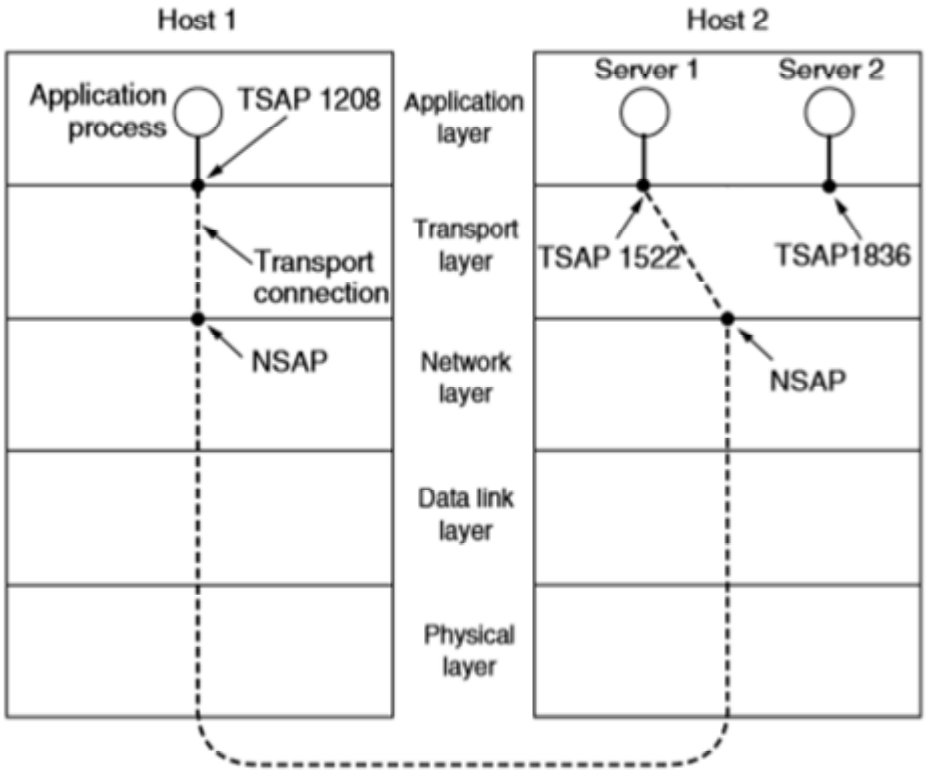


Figure 12.6 TSAPs, NSAPs, and transport connections

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.
3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.

12.4.2 Connection Establishment

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behavior causes serious complications.

Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network uses datagrams inside and that every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost.The crux of the problem is that the delayed duplicates are thought to be new packets. We cannot prevent packets from being duplicated and delayed. But if and when this happens, the packets must be rejected as duplicates and not processed as fresh packets.

Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

The first technique includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the (now known) longest possible path. It is difficult, given that internets may range from a single city to international in scope.

The second method consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.

The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time. This latter method requires

the router clocks to be synchronized, which itself is a nontrivial task, and in practice a hop counter is a close enough approximation to age.

To solve the problem of connection establishment, Tomlinson (1975) introduced the three-way handshake. This could also be seen as a way of how TCP connection is established. TCP stands for Transmission Control Protocol which indicates that it does something to control the transmission of the data in a reliable way.

The process of communication between devices over the internet happens according to the current TCP/IP suite model. The Application layer is a top pile of stack of TCP/IP model from where network referenced application like web browser on the client side establish connection with the server. From the application layer,the information is transferred to the transport layer where our topic comes into picture. The two important protocols of this layer are - TCP, UDP(User Datagram Protocol) out of which TCP is prevalent(since it provides reliability for the connection established). However you can find application of UDP in querying the DNS server to get the binary equivalent of the Domain Name used for the website.

TCP provides reliable communication with something called Positive Acknowledgement with Re-transmission(PAR). The Protocol Data Unit(PDU) of the transport layer is called segment. Now a device using PAR resend the data unit until it receives an acknowledgement. If the data unit received at the receiver's end is damaged(It checks the data with checksum functionality of the transport layer that is used for Error Detection), then receiver discards the segment. So the sender has to resend the data unit for which positive acknowledgement is not received. You can realize from above mechanism that three segments are exchanged between sender(client) and receiver(server) for a reliable TCP connection to get established. The mechanism is shown in the Fig. 12.7.

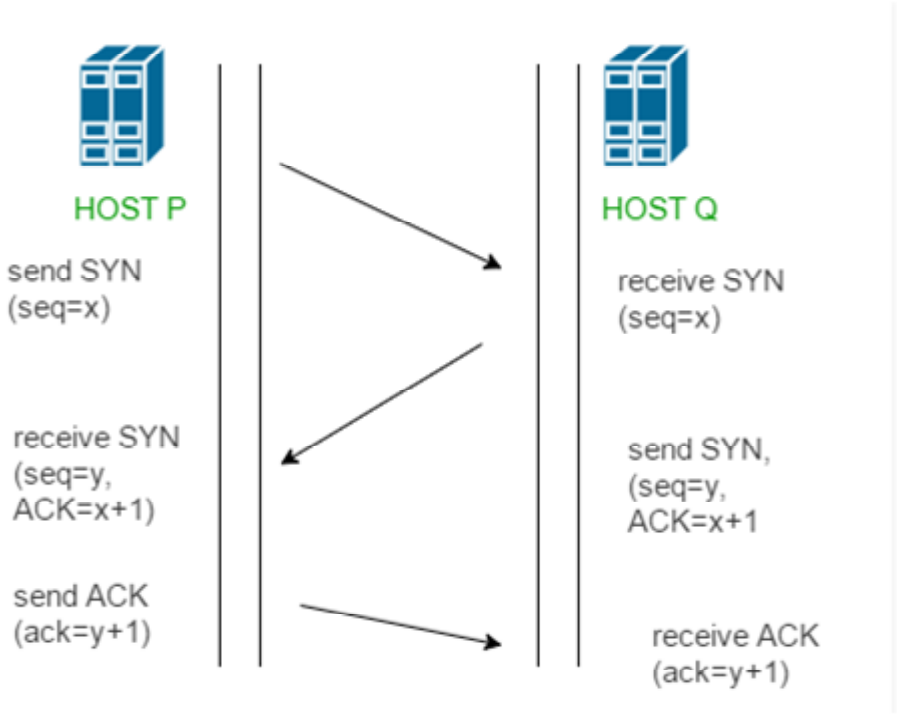


Figure 12.7 3-Way handshake connection establishment

- **Step 1 (SYN) :** In the first step, client wants to establish a connection with server, so it sends a segment with SYN(Synchronize Sequence Number) which informs server that client is likely to start communication with a sequence number.
- **Step 2 (SYN + ACK):** Server responds to the client request with SYN-ACK signal bits set. Acknowledgement(ACK) signifies the response of segment it received and SYN signifies with the sequence number.
- **Step 3 (ACK) :** In the final part client acknowledges the response of server and they both establish a reliable connection with which they will start the actual data transfer.

The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

12.4.3 Connection Release

Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect here. As we mentioned earlier, there are two styles of terminating a connection: asymmetric release and symmetric release.Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken. Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately. Asymmetric release is abrupt and may result in data loss.TCP close connection between Client and Server. Here we will also need to send bit segments to server which FIN bit is set to 1. The connection release is explained in the Fig. 12.8.

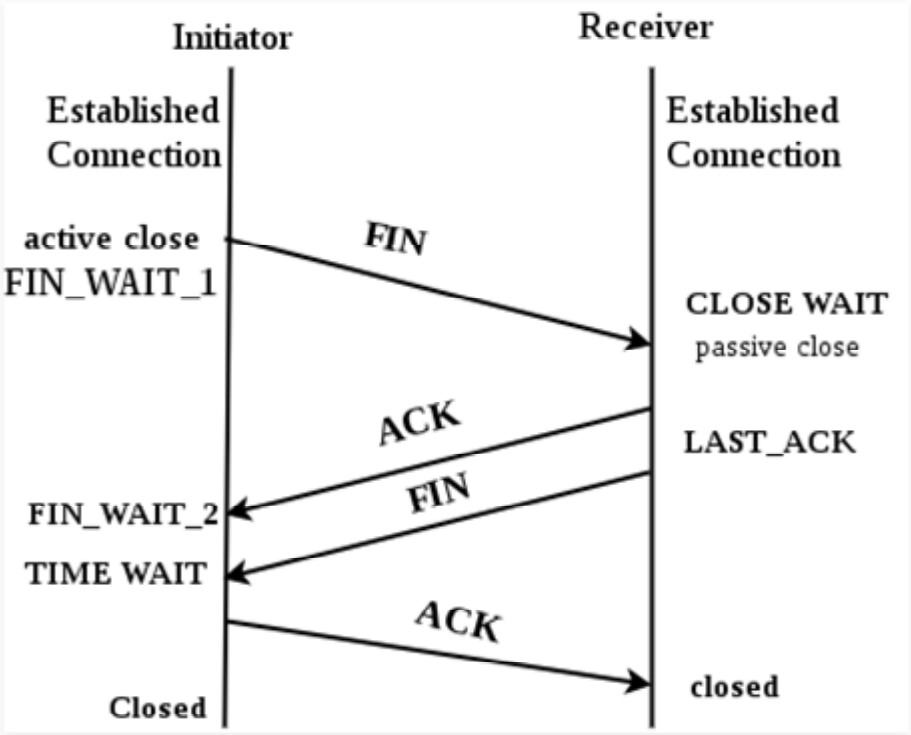


Figure 12.8 Connection release

Now let us explain the above figure.

1. **Step 1** (FIN From Client) - Suppose that the client application decides it wants to close the connection. (Note that the server could also choose to close the connection). This causes the client send a TCP segment with the FIN bit set to 1 to server and to enter the FIN_WAIT_1 state. While in the FIN_WAIT_1 state, the client waits for a TCP segment from the server with an acknowledgment (ACK).

2. **Step 2** (ACK From Server) - When Server received FIN bit segment from Sender (Client), Server Immediately send acknowledgement (ACK) segment to the Sender (Client).
3. **Step 3** (Client waiting) - While in the FIN_WAIT_1 state, the client waits for a TCP segment from the server with an acknowledgment. When it receives this segment, the client enters the FIN_WAIT_2 state. While in the FIN_WAIT_2 state, the client waits for another segment from the server with the FIN bit set to 1.
4. **Step 4** (FIN from Server) - Server sends FIN bit segment to the Sender(Client) after some time when Server send the ACK segment (because of some closing process in the Server).
5. **Step 5** (ACK from Client) - When Client receive FIN bit segment from the Server, the client acknowledges the server's segment and enters the TIME_WAIT state. The TIME_WAIT state lets the client resend the final acknowledgment in case the ACK is lost.The time spent by client in the TIME_WAIT state is depend on their implementation, but their typical values are 30 seconds, 1 minute, and 2 minutes. After the wait, the connection formally closes and all resources on the client side (including port numbers and buffer data) are released.

12.4.4 Error Control and Flow Control

Having examined connection establishment and release in some detail, let us now look at how connections are managed while they are in use. The key issues are error control and flow control. Error control is ensuring that the data is delivered with the desired level of reliability, usually that all of the data is delivered without any errors. Flow control is keeping a fast transmitter from overrunning a slow receiver.

Both of these issues have come up in previous lessons and as a very brief recap:

1. A frame carries an error-detecting code (e.g., a CRC or checksum) that is used to check if the information was correctly received.
2. A frame carries a sequence number to identify itself and is retransmitted by the sender until it receives an acknowledgement of successful receipt from the receiver. This is called ARQ (Automatic Repeat reQuest).

3. There is a maximum number of frames that the sender will allow to be outstanding at any time, pausing if the receiver is not acknowledging frames quickly enough. If this maximum is one packet the protocol is called stop-and-wait. Larger windows enable pipelining and improve performance on long, fast links.
4. The sliding window protocol combines these features and is also used to support bidirectional data transfer.

Given that transport protocols generally use larger sliding windows, we will look at the issue of buffering data more carefully. Since a host may have manyconnections, each of which is treated separately, it may need a substantial amount of buffering for the sliding windows. The buffers are needed at both the sender and the receiver. Certainly they are needed at the sender to hold all transmitted but as yet unacknowledged segments. They are needed there because these segments may be lost and need to be retransmitted.

However, since the sender is buffering, the receiver may or may not dedicate specific buffers to specific connections, as it sees fit. The receiver may, for example, maintain a single buffer pool shared by all connections. When a segment comes in, an attempt is made to dynamically acquire a new buffer. If one is available, the segment is accepted; otherwise, it is discarded. Since the sender is prepared to retransmit segments lost by the network, no permanent harm is done by having the receiver drop segments, although some resources are wasted. The sender just keeps trying until it gets an acknowledgement.

The best trade-off between source buffering and destination buffering depends on the type of traffic carried by the connection. For low-bandwidth bursty traffic, such as that produced by an interactive terminal, it is reasonable not to dedicate any buffers, but rather to acquire them dynamically at both ends, relying on buffering at the sender if segments must occasionally be discarded. On the other hand, for file transfer and other high-bandwidth traffic, it is better if the receiver does dedicate a full window of buffers, to allow the data to flow at maximum speed. This is the strategy that TCP uses.

As connections are opened and closed and as the traffic pattern changes, the sender and receiver need to dynamically adjust their buffer allocations. Consequently, the transport protocol

should allow a sending host to request buffer space at the other end. Buffers could be allocated per connection, or collectively, for all the connections running between the two hosts. Alternatively, the receiver, knowing its buffer situation (but not knowing the offered traffic) could tell the sender "I have reserved X buffers for you." If the number of open connections should increase, it may be necessary for an allocation to be reduced, so the protocol should provide for this possibility.

A reasonably general way to manage dynamic buffer allocation is to decouple the buffering from the acknowledgements, in contrast to the sliding window protocols. Dynamic buffer management means, in effect, a variable-sized window. Initially, the sender requests a certain number of buffers, based on its expected needs. The receiver then grants as many of these as it can afford. Every time the sender transmits a segment, it must decrement its allocation, stopping altogether when the allocation reaches zero. The receiver separately piggybacks both acknowledgements and buffer allocations onto the reverse traffic. TCP uses this scheme, carrying buffer allocations in a header field called Window size.

What is needed is a mechanism that limits transmissions from the sender based on the network's carrying capacity rather than on the receiver's buffering capacity. Belsnes (1975) proposed using a sliding window flow-control scheme in which the sender dynamically adjusts the window size to match the network's carrying capacity. This means that a dynamic sliding window can implement both flow control and congestion control. If the network can handle c segments/sec and the round-trip time (including transmission, propagation, queueing, processing at the receiver, and return of the acknowledgement) is r, the sender's window should be cr. With a window of this size, the sender normally operates with the pipeline full. Any small decrease in network performance will cause it to block. Since the network capacity available to any given flow varies over time, the window size should be adjusted frequently, to track changes in the carrying capacity.

12.4.5 Multiplexing

Multiplexing, or sharing several conversations over connections, virtual circuits, and physical links plays a role in several layers of the network architecture. In the transport layer, the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it. When

a segment comes in, some way is needed to tell which process to give it to. This situation, called multiplexing, is shown in Fig. 12.9(a). In this figure, four distinct transport connections all use the same network connection (e.g., IP address) to the remote host.

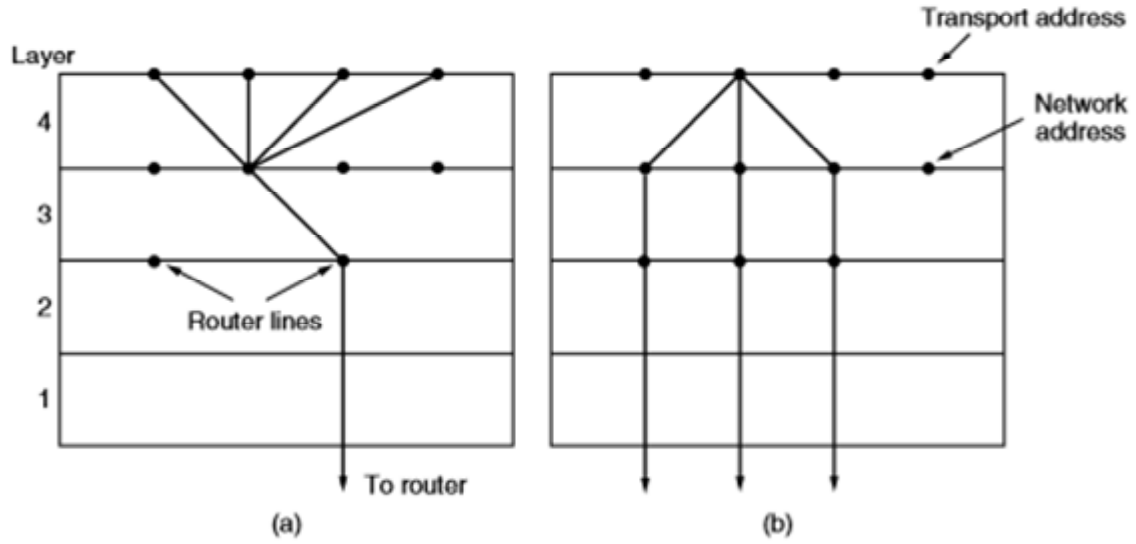


Figure 12.9(a) Multiplexing (b) Inverse multiplexing

Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a host has multiple network paths that it can use. If a user needs more bandwidth or more reliability than one of the network paths can provide, a way out is to have a connection that distributes the traffic among multiple network paths on a round-robin basis, as indicated in Fig. 12.9(b). This modus operandi is called inverse multiplexing. With k network connections open, the effective bandwidth might be increased by a factor of k. An example of inverse multiplexing is SCTP (Stream Control Transmission Protocol), which can run a connection using multiple network interfaces. In contrast, TCP uses a single network endpoint. Inverse multiplexing is also found at the link layer, when several low-rate links are used in parallel as one high-rate link.

12.4.6 Crash Recovery

If hosts and routers are subject to crashes or connections are long-lived (e.g., large software or media downloads), recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. The

transport entities expect lost segments all the time and know how to cope with them by using retransmissions.

A more troublesome problem is how to recover from host crashes. In particular, it may be desirable for clients to be able to continue working when servers crash and quickly reboot. To illustrate the difficulty, let us assume that one host, the client, is sending a long file to another host, the file server, using a simple stop-and-wait protocol. The transport layer on the server just passes the incoming segments to the transport user, one by one. Partway through the transmission, the server crashes. When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was.

In an attempt to recover its previous status, the server might send a broadcast segment to all other hosts, announcing that it has just crashed and requesting that its clients inform it of the status of all open connections. Each client can be in one of two states: one segment outstanding, S1, or no segments outstanding, S0. Based on only this state information, the client must decide whether to retransmit the most recent segment. The client should retransmit if and only if it has an unacknowledged segment outstanding (i.e., is in state S1) when it learns of the crash.

No matter how the client and server are programmed, there are always situations where the protocol fails to recover properly. The server can be programmed in one of two ways: acknowledge first or write first. The client can be programmed in one of four ways: always retransmit the last segment, never retransmit the last segment, retransmit only in state S0, or retransmit only in state S1. This gives eight combinations, but as we shall see, for each combination there is some set of events that makes the protocol fail.

Three events are possible at the server: sending an acknowledgement (A), writing to the output process (W), and crashing (C). The three events can occur in six different orderings: AC(W), AWC, C(AW), C(WA), WAC, and WC(A), where the parentheses are used to indicate that neither A nor W can follow C (i.e., once it has crashed, it has crashed). Figure 12.10 shows all eight combinations of client and server strategies and the valid event sequences for each one. Notice that for each strategy there is some sequence of events that causes the protocol to

fail. For example, if the client always retransmits, the AWC event will generate an undetected duplicate, even though the other two events work properly.

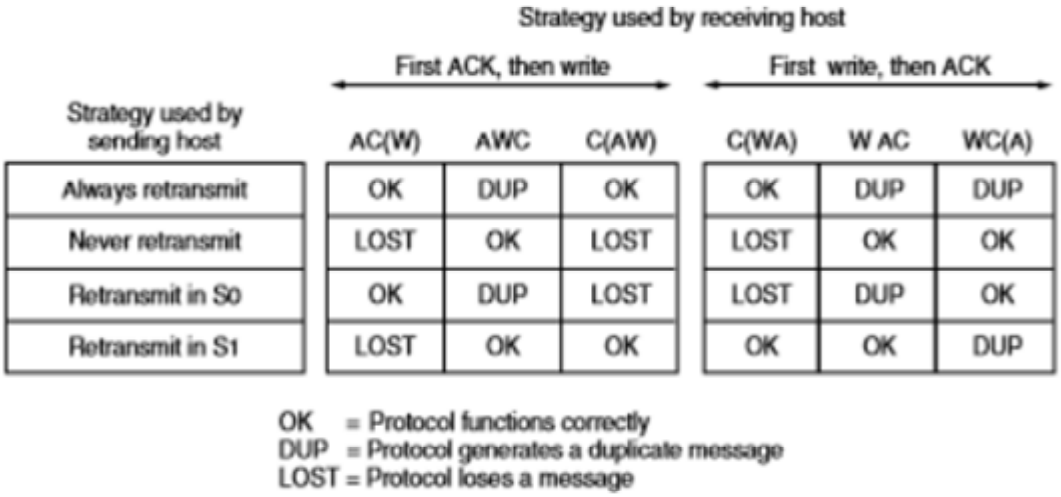


Figure 12.10 Different combinations of client and server strategies

Put in more general terms, this result can be restated as "recovery from a layer N crash can only be done by layer N + 1," and then only if the higher layer retains enough status information to reconstruct where it was before the problem occurred. This is consistent with the case mentioned above that the transport layer can recover from failures in the network layer, provided that each end of a connection keeps track of where it is.

12.5 Summary

The transport layer is the key to understanding layered protocols. It provides various services, the most important of which is an end-to-end, reliable, connection-oriented byte stream from sender to receiver. It is accessed through service primitives that permit the establishment, use, and release of connections. A common transport layer interface is the one provided by Berkeley sockets.

Transport protocols must be able to do connection management over unreliable networks. Connection establishment is complicated by the existence of delayed duplicate packets that can reappear at inopportune moments. To deal with them, three-way handshakes are needed

to establish connections. Releasing a connection is easier than establishing one but is still far from trivial due to the two-army problem. Even when the network layer is completely reliable, the transport layer has plenty of work to do. It must handle all the service primitives, manage connections and timers, allocate bandwidth with congestion control, and run a variable sized sliding window for flow control.

12.6 Model Questions

1. Explain the steps in establishing and releasing a Connection in Transport layer.
2. Discuss on crash recovery. Or (b) Discuss on transposition ciphers.
3. List out the functions of transport layer and describe TCP protocol.
4. How to establish and release a connection? Discuss.
5. Give detailed description about multiplexing.
6. Explain the various classes of addressing.
7. Write notes on multiplexing and crash recovery.
8. Write notes on Transport layer addressing.
9. Explain the elements of Transport Protocols.

LESSON-13

THE INTERNET TRANSPORT PROTOCOL: TCP

Structure

13.1 Introduction

UDP is a simple protocol and it has some very important uses, such as clientserver interactions and multimedia, but for most Internet applications, reliable, sequenced delivery is needed. UDP cannot provide this, so another protocol is required. It is called TCP and is the main workhorse of the Internet. Let us now study it in detail.

13.2 Objectives

- To discuss about the TCP protocol and it's service model and segment header.
- To explain the TCP connection establishment, connection release and congestion control.

13.3 Introduction to TCP

TCP (Transmission Control Protocol) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters. TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.

Each machine supporting TCP has a TCP transport entity, either a library procedure, a user process, or most commonly part of the kernel. In all cases, it manages TCP streams and interfaces to the IP layer. A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64 KB and sends each piece as a separate IP datagram. When datagrams containing TCP data arrive at a machine, they are given to the TCP entity, which reconstructs the original byte streams.

The IP layer gives no guarantee that datagrams will be delivered properly, nor any indication of how fast datagrams may be sent. It is up to TCP to send datagrams fast enough to make use of the capacity but not cause congestion, and to time out and retransmit any datagrams that are not delivered. Datagrams that do arrive may well do so in the wrong order; it is also up to TCP to reassemble them into messages in the proper sequence. In short, TCP must furnish good performance with the reliability that most applications want and that IP does not provide.

13.4 The TCP Service Model

TCP service is obtained by both the sender and the receiver creating end points, called sockets. Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a port. A port is the TCP name for a TSAP. For TCP service to be obtained, a connection must be explicitly established between a socket on one machine and a socket on another machine.

A socket may be used for multiple connections at the same time. In other words, two or more connections may terminate at the same socket. Connections are identified by the socket identifiers at both ends, that is, (socket1, socket2). No virtual circuit numbers or other identifiers are used.

Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users. They are called well-known ports. Other ports from 1024 through 49151 can be registered with IANA for use by unprivileged users, but applications can and do choose their own ports.All TCP connections are full duplex and point-to-point. Full duplex means that traffic can go in both directions at the same time. Point-to-point means that each connection has exactly two end points. TCP does not support multicasting or broadcasting.

A TCP connection is a byte stream, not a message stream. Message boundaries are not preserved end to end. For example, if the sending process does four 512-byte writes to a TCP stream, these data may be delivered to the receiving process as four 512-byte chunks, two 1024-byte chunks, one 2048-byte chunk or some other way. There is no way for the receiver to detect the unit(s) in which the data were written, no matter how hard it tries.

When an application passes data to TCP, TCP may send it immediately or buffer it (in order to collect a larger amount to send at once), at its discretion. However, sometimes the application really wants the data to be sent immediately. To force data out, TCP has the notion of a PUSH flag that is carried on packets. The original intent was to let applications tell TCP implementations via the PUSH flag not to delay the transmission. However, applications cannot literally set the PUSH flag when they send data. Instead, different operating systems have evolved different options to expedite transmission (e.g., TCP NODELAY in Windows and Linux).

For Internet archaeologists, we will also mention one interesting feature of TCP service that remains in the protocol but is rarely used: urgent data. When an application has high priority data that should be processed immediately, for example, if an interactive user hits the CTRL-C key to break off a remote computation that has already begun, the sending application can put some control information in the data stream and give it to TCP along with the URGENT flag. This event causes TCP to stop accumulating data and transmit everything it has for that connection immediately.

13.5 The TCP Protocol

In this section, we will give a general overview of the TCP protocol. A key feature of TCP, and one that dominates the protocol design, is that every byte on a TCP connection has its own

32-bit sequence number. When the Internet began, the lines between routers were mostly 56-kbps leased lines, so a host blasting away at full speed took over 1 week to cycle through the sequence numbers. At modern network speeds, the sequence numbers can be consumed at an alarming rate. Separate 32-bit sequence numbers are carried on packets for the sliding window position in one direction and for acknowledgements in the reverse direction.

The sending and receiving TCP entities exchange data in the form of segments. A TCP segment consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments should be. It can accumulate data from several writes into one segment or can split data from one write over multiple segments. Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515byte IP payload. Second, each link has an MTU (Maximum Transfer Unit). Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, unfragmented packet. In practice, the MTU is generally 1500 bytes and thus defines the upper bound on segment size.

The basic protocol used by TCP entities is the sliding window protocol with a dynamic window size. When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, and otherwise without) bearing an acknowledgement number equal to the next sequence number it expects to receive and the remaining window size. If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

Although this protocol sounds simple, there are many sometimes subtle ins and outs, which we will cover below. Segments can arrive out of order, so bytes 3072-4095 can arrive but cannot be acknowledged because bytes 2048-3071 have not turned up yet. Segments can also be delayed so long in transit that the sender times out and retransmits them. The retransmissions may include different byte ranges than the original transmission, requiring careful administration to keep track of which bytes have been correctly received so far. However, since each byte in the stream has its own unique offset, it can be done.

TCP must be prepared to deal with these problems and solve them in an efficient way. A considerable amount of effort has gone into optimizing the performance of TCP streams, even in the face of network problems.

13.6 The TCP Segment Header

The length of TCP header is minimum 20 bytes long and maximum 60 bytes. Figure 13.1 shows the layout of a TCP segment.

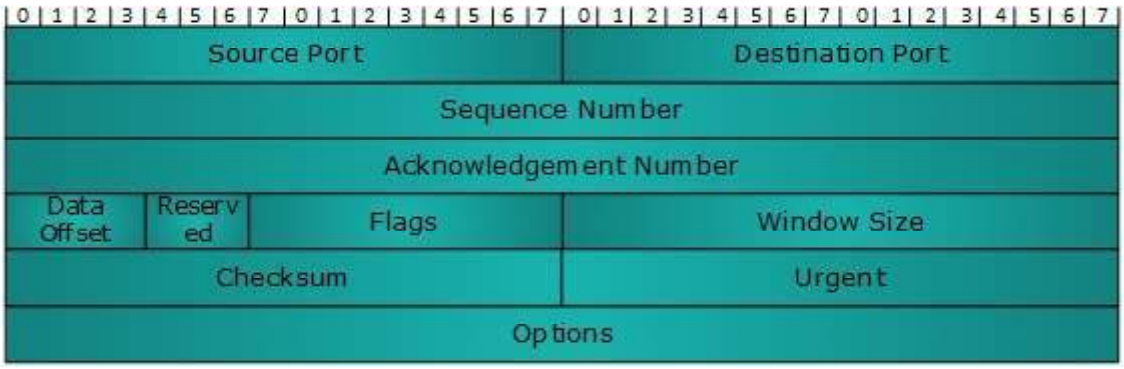


Figure 13.1 The TCP header

- Source Port (16-bits) - It identifies source port of the application process on the sending device.
- Destination Port (16-bits) - It identifies destination port of the application process on the receiving device.
- Sequence Number (32-bits) - Sequence number of data bytes of a segment in a session.
- Acknowledgement Number (32-bits) - When ACK flag is set, this number contains the next sequence number of the data byte expected and works as acknowledgement of the previous data received.
- Data Offset (4-bits) - This field implies both, the size of TCP header (32-bit words) and the offset of data in current packet in the whole TCP segment.
- Reserved (3-bits) - Reserved for future use and all are set zero by default.
- Flags (1-bit each)
    - NS - Nonce Sum bit is used by Explicit Congestion Notification signaling process.
    - CWR - When a host receives packet with ECE bit set, it sets Congestion Windows Reduced to acknowledge that ECE received.
    - ECE -It has two meanings:

- If SYN bit is clear to 0, then ECE means that the IP packet has its CE (congestion experience) bit set.
- If SYN bit is set to 1, ECE means that the device is ECT capable.
    - o URG - It indicates that Urgent Pointer field has significant data and should be processed.
    - o ACK - It indicates that Acknowledgement field has significance. If ACK is cleared to 0, it indicates that packet does not contain any acknowledgement.
    - o PSH - When set, it is a request to the receiving station to PUSH data (as soon as it comes) to the receiving application without buffering it.
    - o RST - Reset flag has the following features:
- It is used to refuse an incoming connection.
- It is used to reject a segment.
- It is used to restart a connection.
    - o SYN - This flag is used to set up a connection between hosts.
    - o FIN - This flag is used to release a connection and no more data is exchanged thereafter. Because packets with SYN and FIN flags have sequence numbers, they are processed in correct order.
- Windows Size - This field is used for flow control between two stations and indicates the amount of buffer (in bytes) the receiver has allocated for a segment, i.e. how much data is the receiver expecting.
- Checksum - This field contains the checksum of Header, Data and Pseudo Headers.
- Urgent Pointer - It points to the urgent data byte if URG flag is set to 1.
- Options - It facilitates additional options which are not covered by the regular header. Option field is always described in 32-bit words. If this field contains data less than 32-bit, padding is used to cover the remaining bits to reach 32-bit boundary.

13.7 TCP Connection Establishment

TCP is a connection oriented protocol and every connection oriented protocol needs to establish connection in order to reserve resources at both the communicating ends.

1. Sender starts the process with following:
    - Sequence number (Seq=521): contains the random initial sequence number which generated at sender side.
    - Syn flag (Syn=1): request receiver to synchronize its sequence number with the above provided sequence number.
    - Maximum segment size (MSS=1460 B): sender tells its maximum segment size, so that receiver sends datagram which won't require any fragmentation. MSS field is present inside Option field in TCP header.
    - Window size (window=14600 B): sender tells about his buffer capacity in which he has to store messages from receiver.
2. TCP is a full duplex protocol so both sender and receiver require a window for receiving messages from one another.
    - Sequence number (Seq=2000): contains the random initial sequence number which generated at receiver side.
    - Syn flag (Syn=1): request sender to synchronize its sequence number with the above provided sequence number.
    - Maximum segment size (MSS=500 B): sender tells its maximum segment size, so that receiver sends datagram which won't require any fragmentation. MSS field is present inside Option field in TCP header.Since MSSreceiver <MSSsender, both parties agree for minimum MSS i.e., 500 B to avoid fragmentation of packets at both ends. Therefore, receiver can send maximum of 14600/500=29 packets. This is the receiver's sending window size.
    - Window size (window=10000 B): receiver tells about his buffer capacity in which he has to store messages from sender. Therefore, sender can send a maximum of 10000/500 = 20 packets. This is the sender's sending window size.

- Acknoledgement Number (Ack no.=522): Since sequence number 521 is received by receiver so, it makes a request of next sequence number with Ack no.=522 which is the next packet expected by receiver since Syn flag consumes 1 sequence no.
- ACK flag (ACk=1): tells that acknowledgement number field contains the next sequence expected by receiver.

3. Sender makes the final reply for connection establishment in following way:
   - Sequence number (Seq=522): since sequence number = 521 in 1st step and SYN flag consumes one sequence number hence, next sequence number will be 522.
   - Acknowledgement Number (Ack no.=2001): since sender is acknowledging SYN=1 packet from the receiver with sequence number 2000 so, the next sequence number expected is 2001.
   - ACK flag (ACK=1): tells that acknowledgement number field contains the next sequence expected by sender.

The above whole process is explained in the Figure 13.2. Since the connection establishment phase of TCP makes use of 3 packets, it is also known as 3-way Handshaking (SYN, SYN + ACK, ACK).

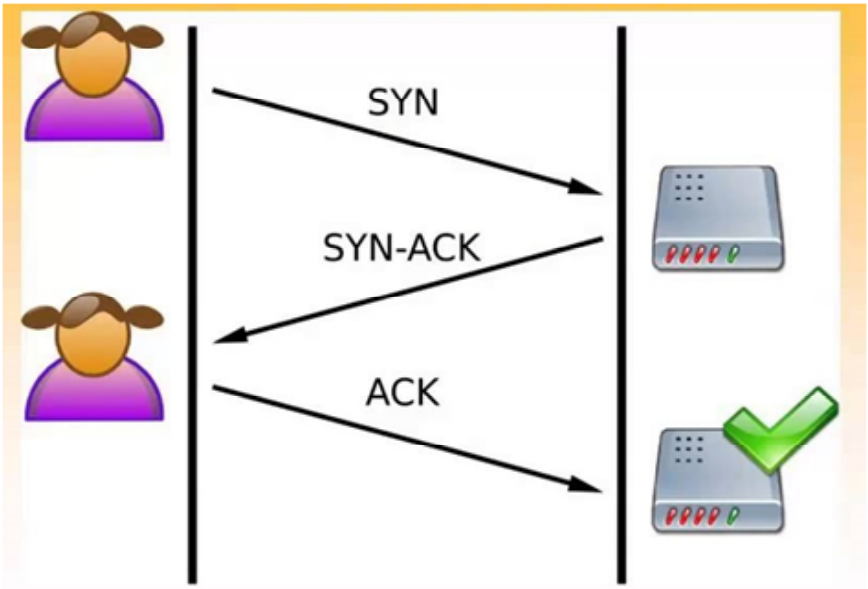


Figure 13.2 TCP Connection establishment

13.8 TCP Connection Release

Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections. Each simplex connection is released independently of its sibling. To release a connection, either party can send a TCP segment with the FIN bit set, which means that it has no more data to transmit. When the FIN is acknowledged, that direction is shut down for new data. Data may continue to flow indefinitely in the other direction, however. When both directions have been shut down, the connection is released. Normally, four TCP segments are needed to release a connection: one FIN and one ACK for each direction. However, it is possible for the first ACK and the second FIN to be contained in the same segment, reducing the total count to three.

Just as with telephone calls in which both people say goodbye and hang up the phone simultaneously, both ends of a TCP connection may send FIN segments at the same time. These are each acknowledged in the usual way, and the connection is shut down. There is, in fact, no essential difference between the two hosts releasing sequentially or simultaneously.

To avoid the two-army problem, timers are used. If a response to a FIN is not forthcoming within two maximum packet lifetimes, the sender of the FIN releases the connection. The other side will eventually notice that nobody seems to be listening to it anymore and will time out as well. While this solution is not perfect, given the fact that a perfect solution is theoretically impossible, it will have to do. In practice, problems rarely arise.

13.9 TCP Timer Management

TCP uses several timers to ensure that excessive delays are not encountered during communications. Several of these timers are elegant, handling problems that are not immediately obvious at first analysis. Each of the timers used by TCP is examined in the following sections, which reveal its role in ensuring data is properly sent from one connection to another.TCP implementation uses four timers.

13.9.1 Retransmission Timer

To retransmit lost segments, TCP uses ReTransmission Timeout (RTO). When TCP sends a segment the timer starts and stops when the acknowledgment is received. If the timer expires

timeout occurs and the segment is retransmitted. To calculate retransmission timeout we first need to calculate the RTT (Round Trip Time). RTT is of three types. (1) Measured RTT(RTTm) - The measured round-trip time for a segment is the time required for the segment to reach the destination and be acknowledged, although the acknowledgment may include other segments. (2) Smoothed RTT(RTTs) - It is the weighted average of RTTm. RTTm is likely to change and its fluctuation is so high that a single measurement cannot be used to calculate RTO. (3) Deviated RTT(RTTd) - Most implementation do not use RTTs alone so RTT deviated is also calculated to find out RTO.

Retransmission Timeout : RTO calculation

The value of RTO is based on the smoothed round-trip time and its deviation. Most implementations use the following formula to calculate the RTO:

Initital value → Original (given in question)

After any measurement → RTO = RTTs + 4 * RTTd

At every retransmission the value of RTO doubles. (RTO(new) = RTO(before retransmission) *2) as per Karn'salgorithm.

13.9.2 Persistence Timer

To deal with a zero-window-size deadlock situation, TCP uses a persistence timer.When the sending TCP receives an acknowledgment with a window size of zero, it starts a persistence timer. When the persistence timer goes off, the sending TCP sends a special segment called a probe. This segment contains only 1 byte of new data. It has a sequence number, but its sequence number is never acknowledged; it is even ignored in calculating the sequence number for the rest of the data. The probe causes the receiving TCP to resend the acknowledgment which was lost.

13.9.3 Keep Alive Timer

A keep alive timer is used to prevent a long idle connection between two TCPs. If a client opens a TCP connection to a server transfers some data and becomes silent the client will

crash. In this case, the connection remains open forever. So a keepalive timer is used. Each time the server hears from a client, it resets this timer. The time-out is usually 2 hours. If the server does not hear from the client after 2 hours, it sends a probe segment. If there is no response after 10 probes, each of which is 75 s apart, it assumes that the client is down and terminates the connection.

13.9.4 Time Wait Timer

This timer is used during TCP connection termination. The timer starts after sending the last ACK for 2nd FIN and closing the connection.After a TCP connection is closed, it is possible for datagrams that are still making their way through the network to attempt to access the closed port. The quiet timer is intended to prevent the just-closed port from reopening again quickly and receiving these last datagrams.

13.10 TCP Congestion Control

When the load offered to any network is more than it can handle, congestion builds up. The Internet is no exception. The network layer detects congestion when queues grow large at routers and tries to manage it, if only by dropping packets. It is up to the transport layer to receive congestion feedback from the network layer and slow down the rate of traffic that it is sending into the network. In the Internet, TCP plays the main role in controlling congestion, as well as the main role in reliable transport. That is why it is such a special protocol.

Transmission Control Protocol (TCP) uses a network congestion-avoidance algorithm that includes various aspects of an Additive Increase Multiplicative Decrease (AIMD) scheme, along with other schemes including slow start and congestion window, to achieve congestion avoidance. The TCP congestion-avoidance algorithm is the primary basis for congestion control in the internet. As per the end-to-end principle, congestion control is largely a function ofinternet hosts, not the network itself. There are several variations and versions of the algorithm implemented in protocol stacks of operating systems of computers that connect to the internet.

13.10.1 Operation

To avoid congestion collapse, TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a congestion window, limiting the total number of

unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called slow start to increase the congestion window after a connection is initialized or after a timeout. It starts with a window, a small multiple of the Maximum Segment Size (MSS) in size. Although the initial rate is low, the rate of increase is very rapid; for every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every Round Trip Time (RTT).

When the congestion window exceeds the slow-start threshold, ssthresh, the algorithm enters a new state, called congestion avoidance. In congestion avoidance state, as long as non-duplicate ACKs are received the congestion window is additively increased by one MSS every round-trip time.

13.10.2 Congestion Window

In TCP, the congestion window is one of the factors that determines the number of bytes that can be sent out at any time. The congestion window is maintained by the sender. Note that this is not to be confused with the sliding window size which is maintained by the receiver. The congestion window is a means of stopping a link between the sender and the receiver from becoming overloaded with too much traffic. It is calculated by estimating how much congestion there is on the link.

When a connection is set up, the congestion window, a value maintained independently at each host, is set to a small multiple of the MSS allowed on that connection. Further variance in the congestion window is dictated by an AIMD approach. This means that if all segments are received and the acknowledgments reach the sender on time, some constant is added to the window size. When the window reaches ssthresh, the congestion window increases linearly at the rate of 1/(congestion window) segment on each new acknowledgement received. The window keeps growing until a timeout occurs. On timeout: (1) Congestion window is reset to 1 MSS (2) ssthresh is set to half the congestion window size before the timeout and (3) slow start is initiated.

A system administrator may adjust the maximum window size limit, or adjust the constant added during additive increase, as part of TCP tuning.The flow of data over a TCP connection

is also controlled by the use of the receive window advertised by the receiver. By comparing its own congestion window with the receive window, a sender can determine how much data it may send at any given time.

13.10.3 Slow Start

Slow start is part of the congestion control strategy used by TCP in conjunction with other algorithms to avoid sending more data than the network is capable of forwarding, that is, to avoid causing network congestion. The algorithm is specified by RFC 5681.

Although the strategy is referred to as slow start, its congestion window growth is quite aggressive, more aggressive than the congestion avoidance phase.[1] Before slow start was introduced in TCP, the initial pre-congestion avoidance phase was even faster.

Slow start begins initially with a congestion window size (CWND) of 1, 2, 4 or 10 MSS. The value for the congestion window size will be increased by one with each acknowledgement (ACK) received, effectively doubling the window size each round-trip time.[c] The transmission rate will be increased by the slow-start algorithm until either a loss is detected, or the receiver's advertised window (rwnd) is the limiting factor, or ssthresh is reached. If a loss event occurs, TCP assumes that it is due to network congestion and takes steps to reduce the offered load on the network. These measurements depend on the exact TCP congestion avoidance algorithm used.

13.11 Summary

Congestion control should allocate all of the available bandwidth between competing flows fairly, and it should track changes in the usage of the network. The AIMD control law converges to a fair and efficient allocation.

The Internet has two main transport protocols: UDP and TCP. UDP is a connectionless protocol that is mainly a wrapper for IP packets with the additional feature of multiplexing and demultiplexing multiple processes using a single IP address. UDP can be used for client-server interactions, for example, using RPC. It can also be used for building real-time protocols such as RTP. The main Internet transport protocol is TCP. It provides a reliable, bidirectional, congestion-controlled byte stream with a 20-byte header on all segments.

13.12 Model Questions

1. Describe TCP Protocol.
2. Give a note on TCP segment header or Internet Protocol header.
3. Explain any two Internet Control protocol.
4. Write notes on timer based connection management.
5. List out the functions of transport layer and describe TCP protocol.

LESSON - 14
NETWORK SECURITY

Structure

14.1 Introduction

Security is a broad topic and it is concerned with people trying to access remote services that they are not authorized to use. Security also deals with the problems of legitimate messages being captured and replayed, and with people later trying to deny that they sent certain messages.

Most security problems are intentionally caused by malicious people trying to gain some benefit, get attention, or harm someone. It involves outsmarting often intelligent, dedicated, and sometimes well funded adversaries. It should also be clear that measures that will thwart casual attackers will have little impact on the serious ones. Police records show that the most damaging attacks are not perpetrated by outsiders tapping a phone line but by insiders bearing a grudge. Security systems should be designed accordingly.

14.2 Objectives

- To discuss about the Cryptography
- To explain the types of ciphers.
- To illustrate the principles of Cryptography.

14.3 Cryptography

Network security problems can be divided roughly into four closely intertwined areas: secrecy, authentication, nonrepudiation, and integrity control. Secrecy, also called confidentiality,

has to do with keeping information out of the grubby little hands of unauthorized users. This is what usually comes to mind when people think about network security. Authentication deals with determining whom you are talking to before revealing sensitive information or entering into a business deal. Nonrepudiation deals with signatures: how do you prove that your customer really placed an electronic order for ten million left-handed doohickeys at 89 cents each when he later claims the price was 69 cents? Or maybe he claims he never placed any order. Finally, integrity control has to do with how you can be sure that a message you received was really the one sent and not something that a malicious adversary modified in transit or concocted.

People authenticate other people by various means, including recognizing their faces, voices, and handwriting. Proof of signing is handled by signatures on letterhead paper, raised seals, and so on. Tampering can usually be detected byhandwriting, ink, and paper experts. None of these options are available electronically. Clearly, other solutions are needed.

14.3.1 Introduction to Cryptography

Cryptography is a method of protecting information and communications through the use of codes so that only those for whom the information is intended can read and process it. Professionals make a distinction between ciphers and codes. A cipher is a character-for-character or bit-for-bit transformation, without regard to the linguistic structure of the message. In contrast, a code replaces one word with another word or symbol.

Historically, four groups of people have used and contributed to the art of cryptography: the military, the diplomatic corps, diarists, and lovers. Of these, the military has had the most important role and has shaped the field over the centuries. Within military organizations, the messages to be encrypted have traditionally been given to poorly paid, low-level code clerks for encryption and transmission. The sheer volume of messages prevented this work from being done by a few elite specialists. The encryption model for a symmetric-key cipher is shown in the Fig. 14.1.

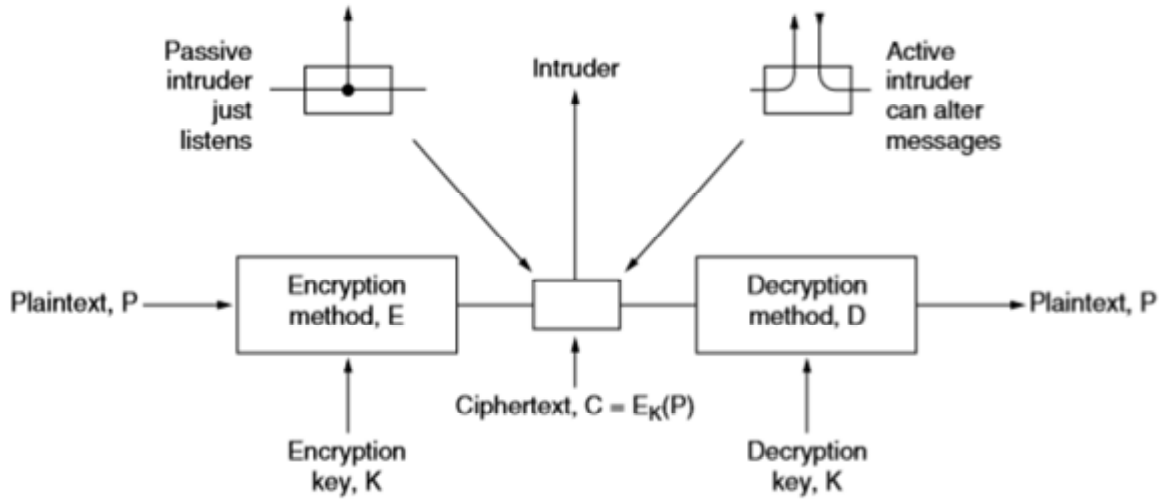


Figure 14.1 The encryption model (for a symmetric-key cipher)

The messages to be encrypted, known as the plaintext, are transformed by a function that is parameterized by a key. The output of the encryption process, known as the ciphertext, is then transmitted, often by messenger or radio. We assume that the enemy, or intruder, hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know what the decryption key is and so cannot decrypt the ciphertext easily. Sometimes the intruder can not only listen to the communication channel (passive intruder) but can also record messages and play them back later, inject his own messages, or modify legitimate messages before they get to the receiver (active intruder). The art ofbreaking ciphers, known as cryptanalysis, and the art of devising them (cryptography) are collectively known as cryptology.

It will often be useful to have a notation for relating plaintext, ciphertext, and keys. We will use C = EK(P) to mean that the encryption of the plaintext P using key K gives the ciphertext C. Similarly, P = DK(C) represents the decryption of C to get the plaintext again. It then follows that

$$DK(EK(P)) = P$$

This notation suggests that E and D are just mathematical functions, which they are. The only tricky part is that both are functions of two parameters, and we have written one of the parameters (the key) as a subscript, rather than as an argument, to distinguish it from the message.

A fundamental rule of cryptography is that one must assume that the cryptanalyst knows the methods used for encryption and decryption. In other words, the cryptanalyst knows how the encryption method, E, and decryption, D, work in detail. The amount of effort necessary to invent, test, and install a new algorithm every time the old method is compromised has always made it impractical to keep the encryption algorithm secret. Thinking it is secret when it is not does more harm than good.

This is where the key enters. The key consists of a (relatively) short string that selects one of many potential encryptions. In contrast to the general method, which may only be changed every few years, the key can be changed as often as required. Thus, our basic model is a stable and publicly known general method parameterized by a secret and easily changed key. The idea that the cryptanalyst knows the algorithms and that the secrecy lies exclusively in the keys is called Kerckhoff's principle. Thus, we have

Kerckhoff's principle: All algorithms must be public; only the keys are secret

Since the real secrecy is in the key, its length is a major design issue. Consider a simple combination lock. The general principle is that you enter digits in sequence. Everyone knows this, but the key is secret. A key length of two digits means that there are 100 possibilities. A key length of three digits means 1000 possibilities, and a key length of six digits means a million. The longer the key, the higher the work factor the cryptanalyst has to deal with.

From the cryptanalyst's point of view, the cryptanalysis problem has three principal variations. When he has a quantity of ciphertext and no plaintext, he is confronted with the ciphertext-only problem. The cryptograms that appear in the puzzle section of newspapers pose this kind of problem. When the cryptanalyst has some matched ciphertext and plaintext, the problem is called the known plaintext problem. Finally, when the cryptanalyst has the ability to encrypt pieces of plaintext of his own choosing, we have the chosen plaintext problem. Newspaper cryptograms could be broken trivially if the cryptanalyst were allowed to ask such questions as "What is the encryption of ABCDEFGHIJKL?"

Encryption methods have historically been divided into two categories: substitution ciphers and transposition ciphers. We will now deal with each of these briefly as background information for modern cryptography.

In a substitution cipher, each letter or group of letters is replaced by another letter or group of letters to disguise it. One of the oldest known ciphers is the Caesar cipher, attributed to Julius Caesar. With this method, a becomes D, b becomes E, c becomes F, ..., and z becomes C. For example, attack becomes DWWDFN. In our examples, plaintext will be given in lowercase letters, and ciphertext in uppercase letters. A slight generalization of the Caesar cipher allows the ciphertext alphabet to be shifted by k letters, instead of always three. In this case, k becomes a key to the general method of circularly shifted alphabets. The Caesar cipher may have fooled Pompey, but it has not fooled anyone since. The next improvement is to have each of the symbols in the plaintext, say, the 26 letters for simplicity, map onto some other letter.

14.3.2 Substitution Ciphers

In a substitution cipher, each letter or group of letters is replaced by another letter or group of letters to disguise it. One of the oldest known ciphers is the Caesar cipher, attributed to Julius Caesar. With this method, a becomes D, b becomes E, c becomes F, ..., and z becomes C. For example, attack becomes DWWDFN. In our examples, plaintext will be given in lowercase letters, and ciphertext in uppercase letters.

A slight generalization of the Caesar cipher allows the ciphertext alphabet to be shifted by k letters, instead of always three. In this case, k becomes a key to the general method of circularly shifted alphabets. The next improvement is to have each of the symbols in the plaintext, say, the 26 letters for simplicity, map onto some other letter. For example,

The general system of symbol-for-symbol substitution is called a monoalphabetic substitution cipher, with the key being the 26-letter string corresponding to the full alphabet. For the key just given, the plaintext attack would be transformed into the ciphertext QZZQEA.

Nevertheless, given a surprisingly small amount of ciphertext, the cipher can be broken easily. The basic attack takes advantage of the statistical properties of natural languages. In English, for example, e is the most common letter, followed by t, o, a, n, i, etc. The most common two-letter combinations, or digrams, are th, in, er, re, and an. The most common three-letter combinations, or trigrams, are the, ing, and, and ion.

A cryptanalyst trying to break a monoalphabetic cipher would start out by counting the relative frequencies of all letters in the ciphertext. Then he might tentatively assign the most common one to e and the next most common one to t. He would then look at trigrams to find a common one of the formtXe, which strongly suggests that X is h. Similarly, if the pattern thYt occurs frequently, the Y probably stands for a. With this information, he can look for a frequently occurring trigram of the form aZW, which is most likely and. By making guesses at common letters, digrams, and trigrams and knowing about likely patterns of vowels and consonants, the cryptanalyst builds up a tentative plaintext, letter by letter.

Another approach is to guess a probable word or phrase. For example, consider the following ciphertext from an accounting firm (blocked into groups of five characters):

```
CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW
```

A likely word in a message from an accounting firm is financial. Using our knowledge that financial has a repeated letter (i), with four other letters between their occurrences, we look for repeated letters in the ciphertext at this spacing. We find 12 hits, at positions 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76, and 82. However, only two of these, 31 and 42, have the next letter (corresponding to n in the plaintext) repeated in the proper place. Of these two, only 31 also has the a correctly positioned, so we know that financial begins at position 30. From this point on, deducing the key is easy by using the frequency statistics for English text and looking for nearly complete words to finish off.

14.3.3 Transposition Ciphers

Substitution ciphers preserve the order of the plaintext symbols but disguise them. Transposition ciphers, in contrast, reorder the letters but do not disguise them. Figure 14.2 depicts a common transposition cipher, the columnar transposition. The cipher is keyed by a word or phrase not containing any repeated letters. In this example, MEGABUCK is the key. The purpose of the key is to order the columns, with column 1 being under the key letter closest to the start of the alphabet, and so on. The plaintext is written horizontally, in rows, padded to fill the matrix if need be. The ciphertext is read out by columns, starting with the column whose key letter is the lowest.

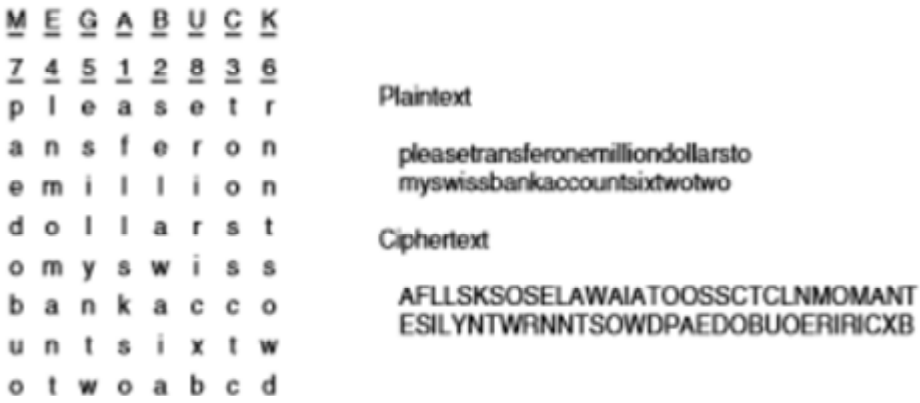


Figure 14.2 A transposition cipher

To break a transposition cipher, the cryptanalyst must first be aware that he is dealing with a transposition cipher. By looking at the frequency of E, T, A, O, I, N, etc., it is easy to see if they fit the normal pattern for plaintext. If so, the cipher is clearly a transposition cipher, because in such a cipher every letter represents itself, keeping the frequency distribution intact.

The next step is to make a guess at the number of columns. In many cases, a probable word or phrase may be guessed at from the context. For example, suppose that our cryptanalyst suspects that the plaintext phrase milliondollars occurs somewhere in the message. Observe that digramsMO, IL, LL, LA, IR, and OS occur in the ciphertext as a result of this phrase wrapping around. The ciphertext letter O follows the ciphertext letter M (i.e., they are vertically adjacent in column 4) because they are separated in the probable phrase by a distance equal to the key

length. If a key of length seven had been used, the digramsMD, IO, LL, LL, IA, OR, andNS would have occurred instead. In fact, for each key length, a different set of digrams is produced in the ciphertext. By hunting for the various possibilities, the cryptanalyst can often easily determine the key length.

The remaining step is to order the columns. When the number of columns, k, is small, each of the k(k – 1) column pairs can be examined in turn to see if its digram frequencies match those for English plaintext. The pair with the best match is assumed to be correctly positioned. Now each of the remaining columns is tentatively tried as the successor to this pair. The column whose digram and trigram frequencies give the best match is tentatively assumed to be correct. The next column is found in the same way. The entire process is continued until a potential ordering is found. Chances are that the plaintext will be recognizable at this point.

Some transposition ciphers accept a fixed-length block of input and produce a fixed-length block of output. These ciphers can be completely described by giving a list telling the order in which the characters are to be output. For example, the cipher of Fig. 14.2 can be seen as a 64 character block cipher. Its output is 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. In other words, the fourth input character, a, is the first to be output, followed by the twelfth, f, and so on.

14.3.4 One-Time Pads

A One Time Pad (OTP) is the only potentially unbreakable encryption method. Plain text encrypted using an OTP cannot be retrieved without the encrypting key. However, there are several key conditions that must be met by the user of a one time pad cipher, or the cipher can be compromised.

- The key must be random and generated by a non-deterministic, non-repeatable process. Any key generated by an algorithm will not work. The security of the OTP relies on the randomness of the key. Unfortunately, the randomness of a key cannot be proved.
- The key must never be reused. Use of the same key to encrypt different messages, no matter how trivially small, compromises the cipher.
- The key must not fall in the hands of the enemy. This may seem obvious, but it points to the weakness of system in that you must be able to transmit large amounts of data to the reader of the pad. Typically, one time pad cipher keys are sent via diplomatic pouch.

A typical one time pad system works like this: Generate a long fresh new random key. XOR the plaintext with the key to create the ciphertext. To decrypt the ciphertext, XOR it with the original key. The system as presented is thus a symmetric and reciprocal cipher. Other functions (e.g., addition modulo n) could be used to combine the key and the plaintext to yield the ciphertext, although the resulting system may not be a reciprocal cipher.

If the key is random and never re-used, an OTP is provably unbreakable. Any ciphertext can be decrypted to any message of the same length by using the appropriate key. Thus, the actual original message cannot be determined from ciphertext alone, as all possible plaintexts are equally likely. This is the only cryptosystem for which such a proof is known.The OTP is extremely simple to implement.

However, there are limitations. Re-use the key and the system becomes extremely weak; it can be broken with pencil and paper. Try to build a "one-time-pad" using some algorithm to generate the keys and you don't have a one-time-pad, you have a stream cipher. There are some very secure stream ciphers, but people who do not know one from a one-time pad are probably not able to design one. It is unfortunately fairly common to see weak stream ciphers advertised as unbreakable one-time pads.

Also, even if you have a well-implemented OTP system and your key is kept secure, consider an attacker who knows the plaintext of part of a message. He can then recover that part of the key and use it to encrypt a message of his own. If he can deliver that instead of yours, you are in deep trouble.

Quantum Cryptography

Interestingly, there may be a solution to the problem of how to transmit the one-time pad over the network, and it comes from a very unlikely source: quantum mechanics.Quantum cryptography is based on the fact that light comes in little packets called photons, which have some peculiar properties. Furthermore, light can be polarized by being passed through a polarizing filter, a fact well known to both sunglasses wearers and photographers. If a beam of light (i.e., a stream of photons) is passed through a polarizing filter, all the photons emerging from it will be polarized in the direction of the filter's axis (e.g., vertically). If the beam is now

passed through a second polarizing filter, the intensity of the light emerging from the second filter is proportional to the square of the cosine of the angle between the axes. If the two axes are perpendicular, no photons get through. The absolute orientation of the two filters does not matter; only the angle between their axes counts.

To generate a one-time pad, Alice needs two sets of polarizing filters. Set one consists of a vertical filter and a horizontal filter. This choice is called a rectilinear basis. A basis (plural: bases) is just a coordinate system. The second set of filters is the same, except rotated 45 degrees, so one filter runs from the lower left to the upper right and the other filter runs from the upper left to the lower right. This choice is called a diagonal basis. Thus, Alice has two bases, which she can rapidly insert into her beam at will. In reality, Alice does not have four separate filters, but a crystal whose polarization can be switched electrically to any of the four allowed directions at great speed. Bob has the same equipment as Alice. The fact that Alice and Bob each have two bases available is essential to quantum cryptography.

For each basis, Alice now assigns one direction as 0 and the other as 1. In the example presented below, we assume she chooses vertical to be 0 and horizontal to be 1. Independently, she also chooses lower left to upper right as 0 and upper left to lower right as 1. She sends these choices to Bob as plaintext.

Now Alice picks a one-time pad, for example based on a random number generator (a complex subject all by itself). She transfers it bit by bit to Bob, choosing one of her two bases at random for each bit. To send a bit, her photon gun emits one photon polarized appropriately for the basis she is using for that bit. For example, she might choose bases of diagonal, rectilinear, rectilinear, diagonal, rectilinear, etc. To send her one-time pad of 1001110010100110 with these bases, she would send the photons. Given the one-time pad and the sequence of bases, the polarization to use for each bit is uniquely determined. Bits sent one photon at a time are called qubits.

Bob does not know which bases to use, so he picks one at random for each arriving photon and just uses it. If he picks the correct basis, he gets the correct bit. If he picks the incorrect basis, he gets a random bitbecause if a photon hits a filter polarized at 45 degrees to

its own polarization, it randomly jumps to the polarization of the filter or to a polarization perpendicular to the filter, with equal probability. This property of photons is fundamental to quantum mechanics. Thus, some of the bits are correct and some are random, but Bob does not know which are which.

14.3.5 Two Fundamental Cryptographic Principles

Although we will study many different cryptographic systems, two principles underlying all of them are important to understand.

Redundancy

The first principle is that all encrypted messages must contain some redundancy, that is, information not needed to understand the message. An example may make it clear why this is needed. Consider a mail-order company, The Couch Potato (TCP), with 60,000 products. Thinking they are being very efficient, TCP's programmers decide that ordering messages should consist of a 16-byte customer name followed by a 3-byte data field (1 byte for the quantity and 2 bytes for the product number). The last 3 bytes are to be encrypted using a very long key known only by the customer and TCP.

At first, this might seem secure, and in a sense it is because passive intruders cannot decrypt the messages. Unfortunately, it also has a fatal flaw that renders it useless. Suppose that a recently fired employee wants to punish TCP for firing her. Just before leaving, she takes the customer list with her. She works through the night writing a program to generate fictitious orders using real customer names. Since she does not have the list of keys, she just puts random numbers in the last 3 bytes, and sends hundreds of orders off to TCP.

When these messages arrive, TCP's computer uses the customers' name to locate the key and decrypt the message. Unfortunately for TCP, almost every 3 byte message is valid, so the computer begins printing out shipping instructions. While it might seem odd for a customer to order 837 sets of children's swings or 540 sandboxes, for all the computer knows, the customer might be planning to open a chain of franchised playgrounds. In this way, an active intruder (the exemployee) can cause a massive amount of trouble, even though she cannot understand the messages her computer is generating.

This problem can be solved by the addition of redundancy to all messages. For example, if order messages are extended to 12 bytes, the first 9 of which must be zeros, this attack no longer works because the ex-employee can no longer generate a large stream of valid messages. The moral of the story is that all messages must contain considerable redundancy so that active intruders cannot send random junk and have it be interpreted as a valid message.

However, adding redundancy makes it easier for cryptanalysts to break messages. Suppose that the mail-order business is highly competitive, and The Couch Potato's main competitor, The Sofa Tuber, would dearly love to know how many sandboxes TCP is selling so it taps TCP's phone line. In the original scheme with 3-byte messages, cryptanalysis was nearly impossible because after guessing a key, the cryptanalyst had no way of telling whether it was right because almost every message was technically legal. With the new 12-byte scheme, it is easy for the cryptanalyst to tell a valid message from an invalid one. Thus, we have

Cryptographic principle 1: Messages must contain some redundancy

In other words, upon decrypting a message, the recipient must be able to tell whether it is valid by simply inspecting the message and perhaps performing asimple computation. This redundancy is needed to prevent active intruders from sending garbage and tricking the receiver into decrypting the garbage and acting on the "plaintext." However, this same redundancy makes it much easier for passive intruders to break the system, so there is some tension here. Furthermore, the redundancy should never be in the form of n0s at the start or end of a message, since running such messages through some cryptographic algorithms gives more predictable results, making the cryptanalysts' job easier. A CRC polynomial is much better than a run of 0s since the receiver can easily verify it, but it generates more work for the cryptanalyst. Even better is to use a cryptographic hash. For the moment, think of it as a better CRC.

Freshness

The second cryptographic principle is that measures must be taken to ensure that each message received can be verified as being fresh, that is, sent very recently. This measure is needed to prevent active intruders from playing back old messages. If no such measures were taken, our ex-employee could tap TCP's phone line and just keep repeating previously sent valid messages.

Cryptographic principle 2: Some method is needed to foil replay attacks

One such measure is including in every message a timestamp valid only for, say, 10 seconds. The receiver can then just keep messages around for 10 seconds and compare newly arrived messages to previous ones to filter out duplicates. Messages older than 10 seconds can be thrown out, since any replays sent more than 10 seconds later will be rejected as too old.

14.4 Summary

Cryptography is a tool that can be used to keep information confidential and to ensure its integrity and authenticity. All modern cryptographic systems are based on Kerckhoff's principle of having a publicly known algorithm and a secretkey. Many cryptographic algorithms use complex transformations involving substitutions and permutations to transform the plaintext into the ciphertext. However, if quantum cryptography can be made practical, the use of one-time pads may provide truly unbreakable cryptosystems.

14.5 Model Questions

1. Discuss the term substitution cipher in cryptography.
2. Discuss on cryptography.
3. Discuss on transposition ciphers.
4. Explain the fundamental principles of Cryptography

Model Question Paper

Master of Computer Applications

Core Paper - XVII

Computer Networks

Time: 3 Hours Maximum :80 marks

PART A - (10 * 2 = 20 Marks)

Answer any TEN questions

1. Write notes on WAN.
2. Define the terms: (i) Gateway (ii) Bridge (iii) Router
3. Write a short notes on Radio Transmission.
4. What do you mean by a framing in Data Link Layer.
5. Write short notes on: Protocol using Go-Back-N.
6. Illustrate: Simplex protocol for a Noisy channel.
7. Define: Collision Free Protocols.
8. What is CSMA and mention it's types.
9. Write short notes on Shortest Path routing.
10. Give a note on TCP segment header or Internet Protocol header.
11. Write notes on timer based connection management.
12. Define: Crash Recovery.

PART B - (5 * 6 = 30 Marks)

Answer any FIVE questions

13. Give a detailed note on wireless networks.

14. Explain in detail about the Internet architecture.

15. Explain the OSI reference model.

16. Explain the steps in establishing and releasing a Connection in Transport layer.

17. Explain the Guided transmission media.

18. Discuss about the design issues of Data Link layer.

19. Describe the Distance vector routing algorithm with an example subnet.

PART C - (3 * 10 = 30 Marks)

Answer any THREE questions

20. Describe OSI and TCP/IP Reference models and compare them.

21. Illustrate Switching.

22. Discuss on Error Detection and Error Correction techniques.

23. Discuss in detail about the Elementary Data Link protocols.

24. Explain any two Internet Control protocol.

———————