NEAR EAST UNIVERSITY

Faculty of Engineering

Depatment of Computer Engineering

INTERNETWORKING WITH TCP / IP PROTOCOL

Graduation Project
COM – 400

Student : Raed Altoom (970208)

Supervisor : Professor
Fahretten Sadigoglu

Lefkoşa – 2001

Acknowledgment

For me, there are two people that indeed deserve every word and feeling of Gratitude, appreciation and thankful, they are my parents - May God bless them and reward them all the best.

I have always been a heavy burden upon them; yet, they never hesitated once in providing me with all the components of success, whether it is financial or spiritual. They have been and are continually providing me with all the Encouragement and support one can have. I would like once again to say "THANKS" For everything dad and mom.

Also I would like acknowledging as well my supervisor Prof. Dr.Fahretten Sadigoglu who gives me desire and corporation to present my project.
Special thanks for all staff of my department of Near East university.
At the last I would like to thanks all my friends, for their enduring and support. I think you will have good time when you read my project.

Abstract

At the past U.S. government agencies have realized the importance and potential of Internet technology for many years and have been funding research that has made possible a global Internet. This project discusses principles and ideas underlying the Internet technology that has resulted from research funded by the Advanced Research Projects Agency (ARPA). The ARPA technology includes a set of network standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and routing traffic. Officially named the TCP/IP Internet Protocol Suite and commonly referred to as TCP/IP (after the names of its two main standards), it can be used to communicate across any set of interconnected networks. For example, some corporations use TCP/IP to interconnect all networks within their corporation, even though the corporation has no connection to outside networks. Other groups use TCP/IP for communication among geographically distant sites.

Although the TCP/IP technology is noteworthy by itself, it is especially interesting because its viability has been demonstrated on a large scale. It forms the base technology for a global Internet that connects homes, university campuses and other schools, corporations, and government labs in 61 countries. In the U.S., The National Science Foundation (NSF), the Department of Energy (DOE), the Department of Defense (DOD), the Health and Human Services Agency, (HHS) and the National Aeronautics and Space Administration (NASA) have all participated in funding the Internet, and use TCP/IP to connect many of their research sites. Known as the ARPA/NSF Internet, the TCP/IP Internet, the global Internet, or just the Internet, the resulting Internet allows researchers at connected institutions to share information with colleagues around the world as easily as they share it with researchers in the next room. An outstanding success, the Internet demonstrates the viability of the TCP/IP technology and shows how it can accommodate a wide variety of underlying network technologies.

List of Abbreviations

ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
ASN	Abstract Syntax Notation
BNC	Bayonet Connector Used for thin Ethernet
DC	Collision Detect
CSMA/CD	Carrier Sense Multiple Access with Collision Detect
DARPA	Defense Advanced Research Projects Agency
DCA	Defense Communications Agency
DNS	Domain Name Service
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
ISO	International Standards Organization (
IRTF	Internet Research Task Force
IETF	Internet Engineering Task Force
IESG	Internet Engineering Steering Group
IP	Internet protocol
LED	light emitting diode
LANs	Local Area Networks
NNI	Network to Network Interface
NFS	Network File System
OSI	Open Systems Interconnect
RIP	Routing Information Protocol
SMTP	Simple Mail Transfer Protocol
SRI-NIC	Stanford Research Institute's Network Information Center
TCP / IP	Transmission Control Protocols & the Internet Protocols
UDP	User Datagram Protocol
UNI	User to Network interfaces
VPI	Virtual Path Identifier
VCI	Virtual Circuit Identifier
WANs	Wide Area Networks

CONTENTS

CHAPTER 1
TCP / IP & THE INTERNET

1.1 INTRODUCTION

In 1969 the Defense Advanced Research Projects Agency (DARPA) funded a research and development project to create an experimental packet switching network. This network called the ARPANET was built to study techniques for providing robust, reliable, vendor – independent data communications. Many techniques of modern data communications were developed in the ARPANET.

The experimantal ARPANET was so succesful that many of the organizations attached to it began to use it for daily communications . In 1975 ARPANET was converted from an experimental network to an operational network , and the responsibility for administering the network was ment given to the Defense Communications Agency (DCA) . The basic TCP / IP (Transmission Control Protocols & the Internet Protocols) protocols were developed after the ARPANET was operational .

About the time that TCP / IP was adopted as a standard , the term Internet came into common usage .

1.2 TCP / IP FEATURES

The popularity of the TCP / IP protocols on the internet did not grow rapidly just because the protocols were there , or because military agnecies mandated their use . They met an important need (world – wide data communications) at the right time , and they had several important features that allowed them to meet this need . These are :

- Open protocol standards , freely available and developed independently from any specific computer hardware or operating system . Because so widely supported , TCP / IP is ideal for uniting different hardware and software , even if you don't communicate on the internet .

- Independence from specific physical network hardware . This allows TCP / IP to integrate many different kinds of networks . TCP / IP can be run over an Ethernet , a token ring , a dial – up line , an X.25 net , and virtually ant other kind of physical transmission media .
- A common addressing scheme that allows any TCP / IP device to uniquely address any other device in the entire network , even if the network is as large as the world – wide Internet .
- Standardized high – level protocols for consistent , widely available for user services .

1.3 PROTOCOL STANDARDS

Protocols are formal rules of behaviour . When computer communicate , it is necessary to define a set of rules to govern their communications .

In data communications these sets of rules are also called protocols . In homogenous networks , a single computer vendor specifies a set of communications rules designed to use the strength of the vendor's operating system and hardware architecture . – TCP / IP attempts to create heterogenous network with open protocols that are independent of operating system and architectural differences . TCP / IP protocols are available to everyone , and are developed and changed bu consensus .

1.4 A DATA COMMUNICATIONS MODEL

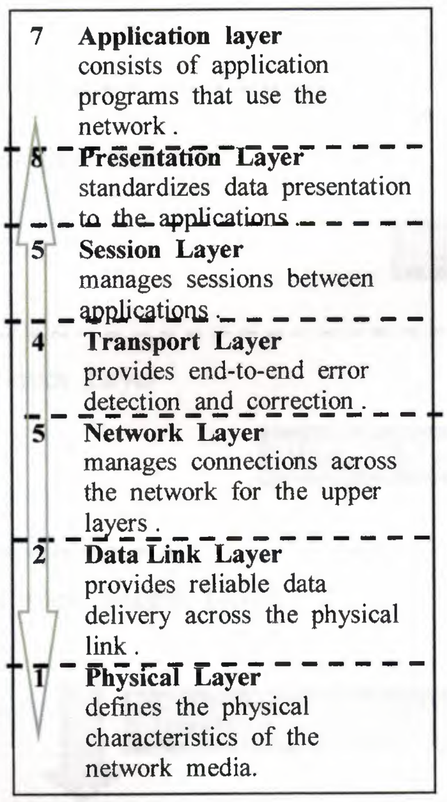
To discuss computer networking , it is necessary to use terms that have special meaning in data communications .

An architectural model developed by the International Standards Organization (ISO) is frequently used to describe the structure and function of data communications protocols . This architectural model , called Open Systems Interconnect (OSI) Reference Model , provides a common reference for discussing communications . The OSI Reference model contains seven layers that define the functions of data communications protocols . Each layer of OSI model represents a function performed when data is transferred between cooperating applications across an intervening network . (figure 1.1)

1.5 TCP / IP PROTOCOL ARCHITECTURE

While there is no universal agreement about how to describew TCP / IP with a layered model , it is generally viewed as being composed of fewer layers than the seven used in the OSI model . Most descriptions of TCP / IP define three to five

functional levels in protocol architecture .



7 **Application layer** consists of application programs that use the network .

8 **Presentation Layer** standardizes data presentation to the applications .

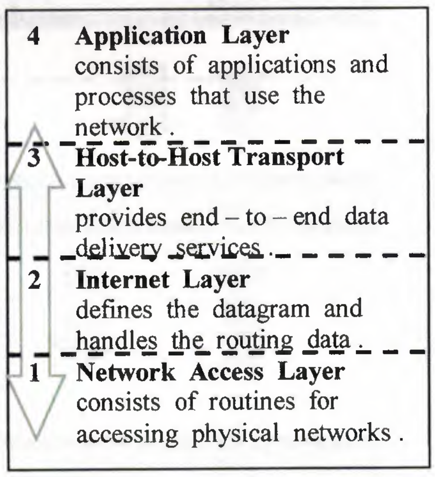
5 **Session Layer** manages sessions between applications .

4 **Transport Layer** provides end-to-end error detection and correction .

5 **Network Layer** manages connections across the network for the upper layers .

2 **Data Link Layer** provides reliable data delivery across the physical link .

1 **Physical Layer** defines the physical characteristics of the network media.



4 **Application Layer** consists of applications and processes that use the network .

3 **Host-to-Host Transport Layer** provides end – to – end data delivery services .

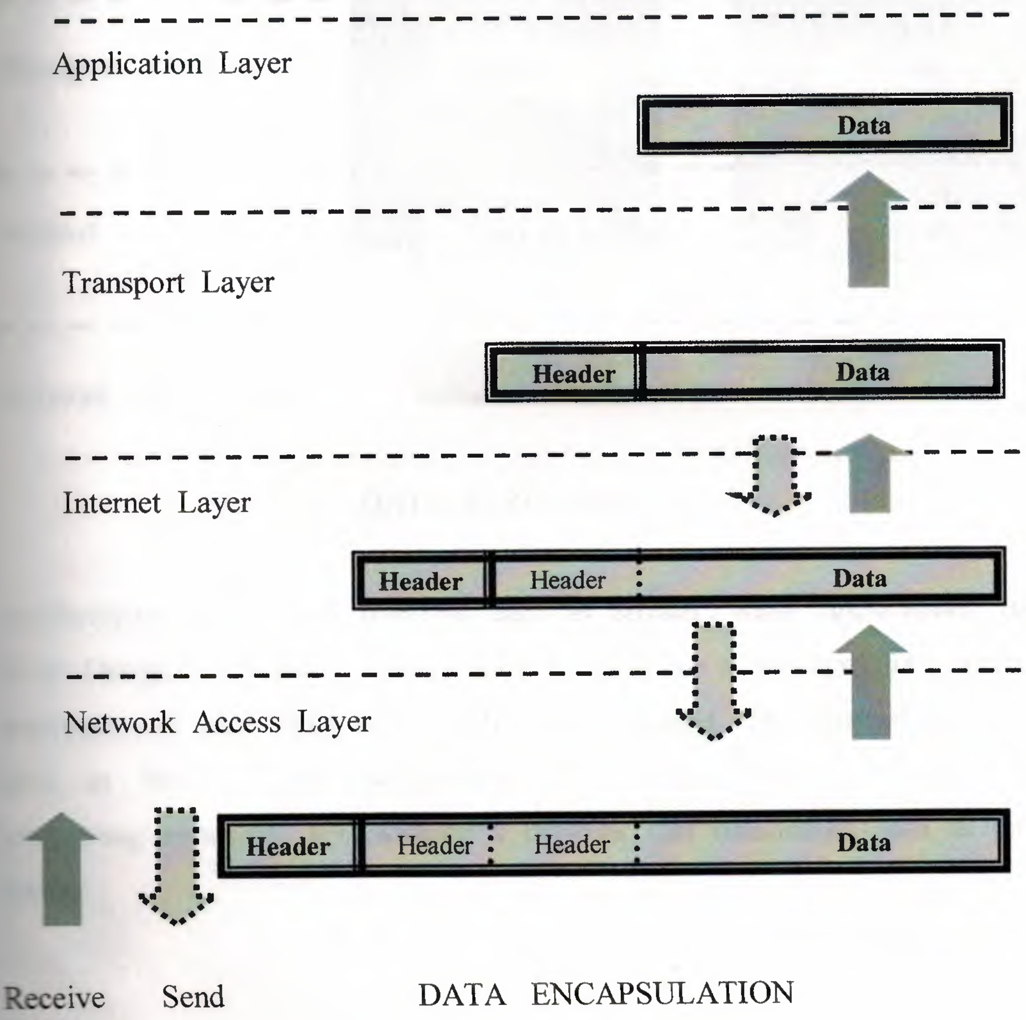
2 **Internet Layer** defines the datagram and handles the routing data .

1 **Network Access Layer** consists of routines for accessing physical networks .

The OSI Reference Model Layers in the TCP/IP Protocol Arch.

FIGURE 1.1 the OSI model

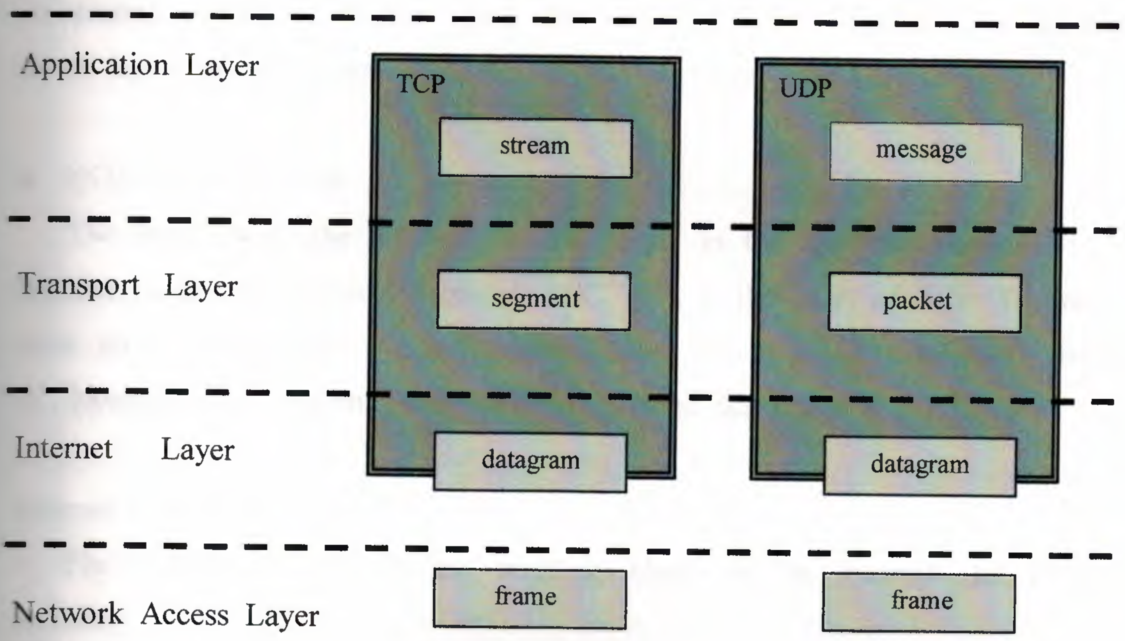
As in the OSI model , data is passed down the stack when it is being sent to the net , and up the stack when it is being received from the network . The four – layered structure of TCP / IP is seen in the way data is handled as it passes down the protocol stack from the Application Layer to the underlying physical network . Each layer in the stack adds control information to ensure proper delivery . This control information is called a header because it is placed in front of the data to be transmitted . Each layer treats all of the information it receives from the layer above as data and places its own header in front of the information . The addition of delivery information at every layer is called encapsulation (figure 1.2) . When data is received , the opposite happens . Each layer strips off its header before passing the data on to the layer above . As information flows back up the stack , information received from a layer is interpreted as both a header and data .



DATA ENCAPSULATION

Each layer has its own independent data structures . Conceptually a layer is unaware of the data structures used by the layers above and below it . In reality , the data structures of a layer are designed to be compatible with the structures used by surrounding layers for take the sake of more efficient data transmission . Still , each layer has its own data structure and its own terminology to describe that structure . You can see the

terms used by different layers of TCP / IP in figure 1.3



DATA STRUCTURES

Applications using TCP refer to data as stream , while applications using the User Datagram Protocol (UDP) refer to data as a message . TCP calls data a segment , and UDP calls its data structure a packet . The internet layer views all data as blocks called datagrams . TCP / IP uses many different types of underlying networks . We asssume a network that transmitted data as packets or frames .

1.6 LAYERS OF TCP / IP

- NETWORK ACCESS LAYER

The Network Access Layer is the lowest layer of the TCP / IP protocol hierarchy .The protocols in this layer provide the means for the system to deliver data to other devices on a directly attached network . It defines how to use the network to transmit an IP datagram . Unlike higher – level protocols , Network Access Layer protocols must know the details of the underlying network to correctly format the data being transmitted to comply with the network constraints . Network Access Layer can encompass the functions of all three lower layers of OSI reference model (Network , Data Link , Physical) .

- INTERNET LAYER

The layer above the Network Access Layer in the protocol hierarchy is the Internet Layer . The Internet Protocol , RFC 791 , is the heart of TCP / IP and the most important protocol in the Internet Layer . All TCP / IP data flows through IP , incoming and outgoing , regardless of its final destination .

Internet Protocol

The Internet protocol is the building block of the internet . Its functions include:

- Defining the datagram , which is the basic unit of transmission in the Internet ;
- Defining the Internet addressing scheme ;
- Moving data between the Network Access Layer and the Host – to – Host Transport Layer ;
- Routing datagrams to remote hosts ;
- Performing fragmentation and re – assembly of datagrams .

IP is a connectionless protocol . This means that IP does not change control information to establish an ent – to – end connection before transmitting data . IP also relies on protocols in other layers to provide error detection and error recovery .

- Internet Control Message Protocol

An integral part of IP is the Internet Control Message Protocol (ICMP) defined in RFC 792 . This protocol is part of the internet layer and uses the IP datagram delivery facility to send its message . ICMP sends messages that perform the following control , error reporting and informational functions for TCP / IP .

  - Flow Control
  - Detecting unreachable destinations
  - Redirecting routes
  - Checking remote host

- TRANSPORT LAYER

The protocol layer just above the Internet Layer is the Host – to – Host Transport Layer . The most important protocols in the Transport Layer are Transmission Comtrol Protocol (TCP) and the User Datagram Protocol (UDP) . TCP provides reliable data delivery service with an end – to – end error detection and correction . UDP provides low – overhead , connectionless datagram deivery service . Both protocols deliver data between the Application Layer and the Internet Layer . Applications programmers can choose whichever service is more appropriate for their specific applications .

- APPLICATION LAYER

At the top of the TCP / IP protocol architecture is the Application Layer . This layer includes all processes that use the Transport Layer protocols to deliver data . There are many applications protocols . Most widely known and implemented applications protocols are :

  - TELNET , the Network Terminal Protocol , provides remote login over the network
  - FTP , the File Transfer Protocol , is used for interactive file transfer .
  - SMTP , the Simple Mail Transfer Protocol delivers electronic mail.

- DNS , Domain Name Service , this application maps IP addresses to the names assigned to network devices .
- RIP , Routing Information Protocol , is central to the way TCP / IP works . It is used by network devices to exchange routing information .
- NFS , Network File System , allows files to be shared by various hosts on the network .

1.7 INTERNET SERVICES

One cannot appreciate the technical details underlying TCP/IP without understanding the services it provides. This section reviews Internet services briefly, highlighting the services most users access, and leaves to later chapters the discussion of how computers connect to a TCP/IP internet and how the functionality is implemented.

Much of our discussion of services will focus on standards called protocols. Protocols like TCP and IP provide the rules for communication. They contain the details of message formats, describe how a computer responds when a message arrives, and specify how a computer handles errors or other abnormal conditions. Most important, they allow us to discuss computer communication independent of any particular vendor's network hardware. In a sense, protocols are to communication what algorithms are to computation. An algorithm allows one to specify or understand a computation without knowing the details of a particular CPU instruction set. Similarly, a communication protocol allows one to specify or understand data communication without depending on detailed knowledge of a particular vendor's network hardware.

Hiding the low-level details of communication helps improve productivity in several ways. First, because programmers deal with higher-level protocol abstractions, they do not need to learn or remember as many details about a given hardware configuration. They can create new programs quickly. Second, because programs built using higher-level abstractions are not restricted to particular machine architecture or particular network hardware, they do not need to be changed when machines or networks are reconfigured. Third, because application programs built using higher-level protocols are independent of the underlying hardware, they can provide direct communion for an arbitrary pair of machines. Programmers do not need to build special versions of application software to move and translate data between each possible pair of chine types.

We will see that all network services are described by protocols. The next sections refer to protocols used to specify application-level services as well as those used to define network-level services. Later chapters explain each of these protocols in more detail.

1.8APPLICATION LEVEL INTERNET SERVICES

From the user is point of view, a TCP/IP Internet appears to be a set of application s that uses the network to carry out useful communication tasks. We use the interoperability to refer to the ability of diverse computing systems to cooperate in solving computational problems. Internet application programs exhibit a high degree of interoperability. Most users that access the Internet do so merely by running application programs without understanding the TCP/IP technology, the structure of the underlying internet, or even the path the data travels to its destination; they rely on the application programs and the underlying network software to handle such details. Only programmers who write network application programs need to view the Internet as a network and need to understand some of the technology. The most popular and widespread Internet application services Include:

- Electronic mail. Electronic mail allows a user to compose memos and send them to individuals or groups. Another part of the mail application allows users to read memos that they have received. Electronic mail has been so successful that many Internet users depend on it for normal business correspondence. Although many electronic mail systems exist, using TCP/IP makes mail delivery more reliable Because it does not rely on intermediate computers to relay mail messages. A TCP/IP mail delivery system operates by having the sender's machine contact the receiver s machine directly. Thus, the sender knows that once the message leaves the local machine, it has been successfully received at the destination site.
- File transfer. Although users sometimes transfer files using electronic mail, mail is designed primarily for short text messages. The TCP/IP protocols include a file transfer application program that allows users to send or receive arbitrarily large

files of programs or data. For example, using the file transfer program, one can copy from one machine to another a large data base containing satellite images, a program written in Pascal or C++, or an English dictionary. The system provides a way to check for authorized users, or even to prevent all access. Like mail, file transfer across a TCP/IP Internet is reliable because the two machines involved communicate directly, without relying on intermediate machines to make copies of the file along the way.

- Remote login. Remote login allows a user sitting at one computer to connect to a remote machine and establish an interactive login session. The remote login makes it appear that a window on the user's screen connects directly to the remote machine by sending each keystroke from the user's keyboard to the remote machine and displaying each character the remote computer prints in the user's window. When the remote login session terminates, the application returns the user to the local system.

We will return to these and other applications in later chapters to examine them in more detail. We will see exactly how they use the underlying TCP/IP protocols, and why having standards for application protocols has helped ensure that they are widespread.

1.9 THE IAB REORGANIZATION

By the summer of 1989, both the TCP/IP technology and the Internet had grown beyond the initial research project into production facilities on which thousands of people depended for daily business. It was no longer possible to introduce new ideas by changing a few installations overnight. To a large extent, the literally hundreds of commercial companies that offer TCP/IP products determined whether products would intemperate by deciding when to incorporate changes in their software. Researchers who drafted specifications and tested new ideas in laboratories could no longer expect instant acceptance and use of the ideas. It was ironic that the researchers who designed and watched TCP/IP develop found themselves overcome by the commercial success of their brainchild. In short, TCP/IP became a successful, production technology and the market place began to dominate its evolution.

To reflect the political and commercial realities of both TCP/IP and the Internet, the

IAB was reorganized in the summer of 1989. The chairmanship changed. Researchers were moved from the IAB itself to a subsidiary group and a new IAB board was constituted to include representatives from the wider community.

Figure 1.1 illustrates the new IAB organization and the relationship of subgroups.

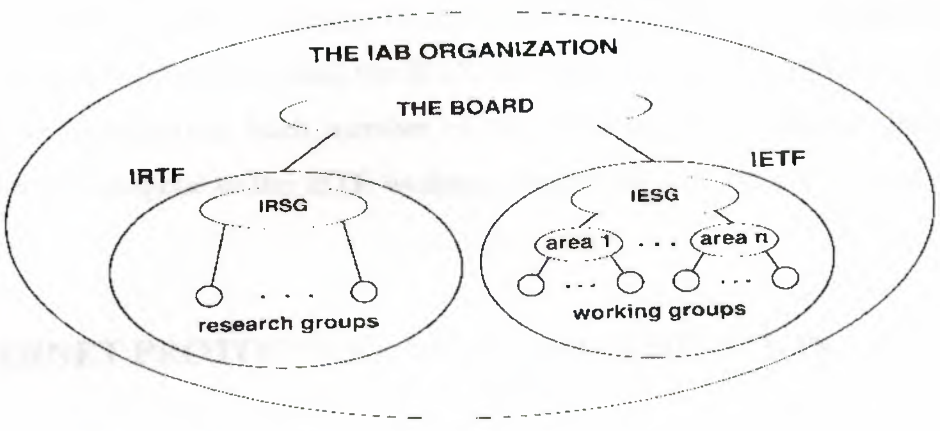


Figure 1.1 the structure of the IAB after the 1989 reorganization.

As Figure 1.1 shows, in addition to the board itself, the IAB organization contains two major groups: the Internet Research Task Force (IRTF) and the Internet Engineering Task Force (IETF).

As its name implies, the IETF concentrates on short-term or medium-term engineering problems. The IETF existed in the original IAB structure, and its success provided part of the motivation for reorganization. Unlike most IAB task forces, which were limited to a few individuals who focused on one specific issue, the IETF grew to include dozens of active members who worked on many problems concurrently. Before the reorganization, the IETF was divided into over 20 working groups, each focusing on a specific problem. Working groups held individual meetings to formulate problem Solutions. In addition, the entire IETF met regularly to hear reports from working groups and discuss proposed changes or additions to the TCP/IP technology. Usually held three times annually, full IETF meetings attracted hundreds of participants and spectators. The IETF had become too large for the chairman to manage.

Because the IETF was known throughout the Internet, and because its meetings were widely recognized and attended, the reorganized IAB structure retains the IETF, but splits it into approximately a dozen areas, each with its own manager. The IETF chairman and the area managers comprise the Internet Engineering Steering Group (IESG), the individuals responsible for coordinating the efforts of IETF working group.

The name "IETF" now refers to the entire body, including the chairman, area managers, and all members of working groups.

Created during the reorganization, the Internet Research Task Force is the research counterpart to the IETF. The IRTF coordinates research activities related to TCP/IP protocols or Internet architecture in general. Like the IETF, the IRTF has a small group called the Internet Research Steering Group or IRSG, which sets priorities and coordinates research activities. Unlike the IETF, the IRTF is currently a much smaller and less active organization. Each member of the IRSG chairs a volunteer Internet Research Group analogous to the IETF working groups; the IRTF is not divided into areas.

1.10 INTERNET PROTOCOLS AND STANDARDIZATION

Readers familiar with data communication networks realize that many communication protocol standards exist. Many of them precede the Internet, so the question arises, "Why did the Internet designers invent new protocols when so many international standards already existed?" The answer is complex, but follows a simple maxim:

Use existing protocol standards whenever such standards apply; invent new protocols only when existing standards are insufficient, and be prepared to use new standards when they become available and provide equivalent functionality.

1.11 GENERAL DESCRIPTION OF THE TCP/IP PROTOCOLS

TCP/IP is a layered set of protocols. In order to understand what this means, it is useful to look at an example. A typical situation is sending mail. First, there is a protocol for mail. This defines a set of commands which one machine sends to another, e.g. commands to specify who the sender of the message is, who it is being sent to, and then the text of the message. However this protocol assumes that there is a way to communicate reliably between the two computers. Mail, like other application protocols, simply defines a set of commands and messages to be sent. It is designed to be used together with TCP and IP. TCP is responsible for making sure that the commands get through to the other end. It keeps track of what is sent, and retransmits

Anything that did not get through. If any message is too large for one Datagram, e.g. the text of the mail, TCP will split it up into several Datagram, and make sure that they all arrive correctly. Since these functions are needed for many applications, they are put together into a separate protocol, rather than being part of the specifications for sending mail. You can think of TCP as forming a library of routines that applications can use when they need reliable network communications with another computer. Similarly, TCP calls on the services of IP. Although the services that TCP supplies are needed by many applications, there are still some kinds of applications that don't need them. However there are some services that every application needs. So these services are put together into IP. As with TCP, you can think of IP as a library of routines that TCP calls on, but which is also available to applications that don't use TCP. This strategy of building several levels of protocol is called "layering". We think of the applications programs such as mail, TCP, and IP, as being separate "layers", each of which calls on the services of the layer below it. Generally, TCP/IP applications use 4 layers:

- an application protocol such as mail
- a protocol such as TCP that provides services need by many applications
- IP, which provides the basic service of getting datagrams to their destination
- The protocols needed to manage a specific physical medium, such as Ethernet or a point to point line.

TCP/IP is based on the "catenet model". (This is described in more detail in IEN 48.) This model assumes that there are a large number of independent networks connected together by gateways. The user should be able to access computers or other resources on any of these networks. Datagrams will often pass through a dozen different networks before getting to their final destination. The routing needed to accomplish this should be completely invisible to the user. As far as the user is concerned, all he needs to know in order to access another system is an "Internet address". This is an address that looks like 128.6.4.194. It is actually a 32-bit number. However it is normally written as 4 decimal numbers, each representing 8 bits of the address. (The term "octet" is used by Internet documentation for such 8-bit chunks. The term "byte" is not used, because some computers that have byte sizes other than 8 bits support TCP/IP.) Generally the structure of the address gives you some information about how to get to the system. For example, 128.6 are a network number assigned by a central authority to Rutgers University. Rutgers uses the next octet to indicate which of the campus Ethernet is involved. 128.6.4 happens to be an Ethernet used by the Computer Science Department. The last octet allows for up to 254 systems on each Ethernet. (It is 254 because 0 and 255 are not allowed, for reasons that will be discussed later.) Note that 128.6.4.194 and 128.6.5.194 would be different systems. The structure of an Internet address is described in a bit more detail later.

Of course we normally refer to systems by name, rather than by Internet address. When we specify a name, the network software looks it up in a database, and comes up with the corresponding Internet address. Most of the network software deals strictly in terms of the address. (RFC 882 describes the name server technology used to handle this lookup.)

TCP/IP is built on "connectionless" technology. Information is transferred as a sequence of "datagrams". A Datagram is a collection of data that is sent as a single message. Each of these datagrams is sent through the network individually. There are provisions to open connections (i.e. to start a conversation that will continue for some time). However at some level, information from those connections is broken up into datagrams, and the network treats those datagrams as completely separate. For example, suppose you want to transfer a 15000-octet file. Most networks can't handle a 15000 octet Datagram. So the protocols will break this up into something like 30 500-octet datagrams. Each of these datagrams will be sent to the other end. At that point, they will be put back together into the 15000-octet file. However while those datagrams are in transit, the network doesn't know that there is any connection between them. It is perfectly possible that Datagram 14 will actually arrive before datagram 13. It is also possible that somewhere in the network, an error will occur, and some datagram won't get through at all. In that case, that datagram has to be sent again.

Note by the way that the term's "datagram" and “packet” often seems to be nearly interchangeable. Technically, datagram is the right word to use when describing TCP/IP. A datagram is a unit of data, which is what the protocols deal with. A packet is a physical thing, appearing on an Ethernet or some wire. In most cases a packet simply contains a datagram, so there is very little difference. However they can differ. When TCP/IP is used on top of X.25, the X.25 interface breaks the datagrams up into 128-byte packets. This is invisible to IP, because the packets are put back together into a single datagram at the other end before being processed by TCP/IP. So in this case, several packets would carry one IP datagram. However with most media, there are efficiency advantages to sending one datagram per packet, and so the distinction tends to vanish.

- The TCP level

Two separate protocols are involved in handling TCP/IP datagrams. TCP (the "transmission control protocol") is responsible for breaking up the message into datagrams, reassembling them at the other end, resenting anything that gets lost, and putting things back in the right order. IP (the "Internet protocol") is responsible for routing individual datagrams. It may seem like TCP is doing all the work. And in small networks that is true. However in the Internet, simply getting a datagram to its destination can be a complex job. A connection may require the datagram to go through several networks at Rutgers, a serial line to the John Von Neuron Supercomputer Center, a couple of Ethernet there, a series of 56Kbaud phone lines to another NSFnet site, and more Ethernet on another campus. Keeping track of the routes to all of the *destinations and handling incompatibilities among different transport media turns out to* be a complex job. Note that the interface between TCP and IP is fairly simple. TCP simply hands IP a datagram with a destination. IP doesn't know how this datagram relates to any datagram before it or after it.

It may have occurred to you that something is missing here. We have talked about Internet addresses, but not about how you keep track of multiple connections to a given system. Clearly it isn't enough to get a datagram to the right destination. TCP has to know which connection this datagram is part of. This task is referred to as "demultiplexing." In fact, there are several levels of demultiplexing going on in TCP/IP. The information needed to do this demultiplexing is contained in a series of "headers". A header is simply a few extra octets tacked onto the beginning of a datagram by some protocol in order to keep track of it. It's a lot like putting a letter into an envelope and putting an address on the outside of the envelope. Except with modern networks it happens several times. It's like you put the letter into a little envelope, your secretary puts that into a somewhat bigger envelope, the campus mail center puts that envelope into a still bigger one, etc. Here is an overview of the headers that get stuck on a message that passes through a typical TCP/IP network:

We start with a single data stream, say a file you are trying to send to some other computer:

TCP breaks it up into manageable chunks. (In order to do this; TCP has to know how large a datagram your network can handle. Actually, the PCP's at each end say how big a datagram they can handle, and then they pick the smallest size.)

TCP puts a header at the front of each datagram. This header actually contains at least 20 octets, but the most important ones are a source and destination "port number" and a "sequence number". The port numbers are used to keep track of different conversations. Suppose 3 different people are transferring files. Your TCP might allocate port numbers 1000, 1001, and 1002 to these transfers. When you are sending a datagram, this becomes the "source" port number, since you are the source of the datagram. Of course the TCP at the other end has assigned a port number of its own for the conversation. Your TCP has to know the port number used by the other end as well. (It finds out when the connection starts, as we will explain below.) It puts this in the "destination" port field. Of course if the other end sends a datagram back to you, the source and destination port numbers will be reversed, since then it will be the source and you will be the destination. Each datagram has a sequence number. This is used so that the other end can make sure that it gets the datagrams in the right order, and that it hasn't missed any. (See the TCP specification for details.) TCP doesn't number the datagrams, but the octets. So if there are 500 octets of data in each datagram, the first datagram might be numbered 0, the second 500, the next 1000, the next 1500, etc. Finally, I will mention the Checksum. This is a number that is computed by adding up all the octets in the datagram (more or less - see the TCP spec). The result is put in the header. TCP at the other end computes the checksum again. If they disagree, then something bad happened to the datagram in transmission, and it is thrown away. So here's what the datagram looks like now.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source Port        | Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Acknowledgment Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Data |          |U|A|P|R|S|F|                       |
| Offset Reserved |R|C|S|S|Y|I|          Window            |
|      |          |G|K|H|T|N|N|                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Checksum          |       Urgent Pointer       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| your data ... next 500 octets              |
| ......                         |
```

If we abbreviate the TCP header as "T", the whole file now looks like this:

```
T.... T.... T.... T.... T.... T.... T....
```

You will note that there are items in the header that I have not described above. They are generally involved with managing the connection. In order to make sure the datagram has arrived at its destination, the recipient has to send back an "acknowledgement". This is a datagram whose "Acknowledgement number" field is filled in. For example, sending a packet with an acknowledgement of 1500 indicates that you have received all the data up to octet number 1500. If the sender doesn't get an acknowledgement within a reasonable amount of time, it sends the data again. The window is used to control how much data can be in transit at any one time. It is not practical to wait for each datagram to be acknowledged before sending the next one. That would slow things down too much. On the other hand, you can't just keep sending, or a fast computer might overrun the capacity of a slow one to absorb data. Thus each end indicates how much new data it is currently prepared to absorb by putting the number of octets in its "Window" field. As the computer receives data, the amount of space left in its window decreases. When it goes to zero, the sender has to stop. As the receiver processes the data, it increases its window, indicating that it is ready to accept more data. Often the same datagram can be used to acknowledge receipt of a set of data and to give permission for additional new data (by an updated window). The "Urgent" field allows one end to tell the other to skip ahead in its processing to a particular octet. This is often useful for handling asynchronous events, for example when you type a control character or other command that interrupts output. The other fields are beyond the scope of this document.

The IP level

TCP sends each of these datagrams to IP. Of course it has to tell IP the Internet address of the computer at the other end. Note that this is all IP is concerned about. It doesn't care about what is in the datagram, or even in the TCP header. IP's job is simply to find a route for the datagram and get it to the other end. In order to allow gateways or other intermediate systems to forward the datagram, it adds its own header. The main things in this header are the source and destination Internet address (32-bit addresses, like 128.6.4.194), the protocol number, and another checksum. The source Internet address is simply the address of your machine. (This is necessary so the other end knows where the datagram came from.) The destination Internet address is the address of the other machine. (This is necessary so any gateways in the middle know where you want the datagram to go.) The protocol number tells IP at the other end to send the datagram to TCP. Although most IP traffic uses TCP, there are other protocols that can use IP, so you have to tell IP which protocol to send the datagram to. Finally, the checksum allows IP at the other end to verify that the header wasn't damaged in transit. Note that TCP and IP have separate checksums. IP needs to be able to verify that the header didn't get damaged in transit, or it could send a message to the wrong place. For reasons not worth discussing here, it is both more efficient and safer to have TCP compute a separate checksum for the TCP header and data. Once IP has tacked on its header, here's what the message looks like:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| IHL |Type of Service|        Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Identification       |Flags|   Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Time to Live |   Protocol   |       Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Source Address                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Destination Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| TCP header, then your data ......                      |
|                                               |
```

If we represent the IP header by an "I", your file now looks like this:

IT.... IT.... IT.... IT.... IT.... IT.... IT....

Again, the header contains some additional fields that have not been discussed. Most of them are beyond the scope of this document. The flags and fragment offset are used to keep track of the pieces when a datagram has to be split up. This can happen when datagrams are forwarded through a network for which they are too big. (This will be discussed a bit more below.) The time to live is a number that is decremented whenever the datagram passes through a system. When it goes to zero, the datagram is discarded.

This is done in case a loop develops in the system somehow. Of course this should be impossible, but well-designed networks are built to cope with "impossible" conditions. At this point, it's possible that no more headers are needed. If your computer happens to have a direct phone line connecting it to the destination computer, or to a gateway, it may simply send the datagrams out on the line (though likely a synchronous protocol such as HDLC would be used, and it would add at least a few octets at the beginning and end).

The Ethernet level

However most of our networks these days use Ethernet. So now we have to describe Ethernet's headers. Unfortunately, Ethernet has its own addresses. The people who designed Ethernet wanted to make sure that no two machines would end up with the same Ethernet address. Furthermore, they didn't want the user to have to worry about assigning addresses. So each Ethernet controller comes with an address built-in from the factory. In order to make sure that they would never have to reuse addresses, the Ethernet designers allocated 48 bits for the Ethernet address. People who make Ethernet equipment have to register with a central authority, to make sure that the numbers they assign don't overlap any other manufacturer. Ethernet is a "broadcast medium". That is, it is in effect like an old party line telephone. When you send a packet out on the Ethernet, every machine on the network sees the packet. So something is needed to make sure that the right machine gets it. As you might guess, this involves the Ethernet header. Every Ethernet packet has a 14-octet header that includes the source and destination Ethernet address, and a type code. Each machine is supposed to pay attention only to packets with its own Ethernet address in the destination field. (It's perfectly possible to cheat, which is one reason that Ethernet communications are not terribly secure.) Note that there is no connection between the Ethernet address and the Internet address. Each machine has to have a table of what Ethernet address corresponds to what Internet address. (We will describe how this table is constructed a bit later.) In addition to the addresses, the header contains a type code. The type code is to allow for several different protocol families to be used on the same network. So you can use TCP/IP, DECnet, Xerox NS, etc. at the same time. Each of them will put a different value in the type field. Finally, there is a checksum. The Ethernet controller computes a checksum of the entire packet. When the other end receives the packet, it recomputes the checksum, and throws the packet away if the answer disagrees with the original. The checksum is put on the end of the packet, not in the header. The final result is that your message looks like this:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Ethernet destination address (first 32 bits)           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Ethernet dest (last 16 bits)  |Ethernet source (first 16 bits)|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Ethernet source address (last 32 bits)                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Type code               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | IP header, then TCP header, then your data                    |
 |                                             |
 | end of your data                                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   Ethernet Checksum                         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If we represent the Ethernet header with "E", and the Ethernet checksum with "C", your file now looks like this:

EIT....C EIT....C EIT....C EIT....C EIT....C

When the other end receives these packets, of course all the headers are removed. The Ethernet interface removes the Ethernet header and the checksum. It looks at the type code. Since the type code is the one assigned to IP, the Ethernet device driver passes the datagram up to IP. IP removes the IP header. It looks at the IP protocol field. Since the protocol type is TCP, it passes the datagram up to TCP. TCP now looks at the sequence number. It uses the sequence numbers and other information to combine all the datagrams into the original file.

The ends our initial summary of TCP/IP. There are still some crucial concepts we haven't gotten to, so we'll now go back and add details in several areas. (For detailed descriptions of the items discussed here see RFC 793 for TCP, RFC 791 for IP, and RFC's 894 and 826 for sending IP over Ethernet.)

Chapter2
THE UNDERLYING NETWORK TECHNOLOGIES

2.1 INTRODUCTION

It is important to understand that the Internet is not a new kind of physical network. It is, instead, a method of interconnecting physical networks and a set of conventions for using networks that allow the computers they reach to interact. While network hardware plays only a minor role in the overall design, understanding the internet technology requires one to distinguish between the low-level mechanisms provided by the hardware itself and the higher-level facilities that the TCP/IP protocol software provides. It is also important to understand how the facilities supplied by packet-switched Technology affects our choice of high-level abstractions.

This chapter introduces basic packet-switching concepts and terminology, and then reviews some of the underlying network hardware technologies that have been used in TCP/IP Internets. Later chapters describe how these networks are interconnected and how the TCP/IP protocols accommodate vast differences in the hardware. While the list presented here is certainly not comprehensive, it clearly demonstrates the variety among physical networks over which TCP/IP operates. The reader can safely skip many of the Technical details, but should try to grasp the idea of packet switching and try to imagine building a homogeneous communication system using such heterogeneous hardware. Most important, the reader should look closely at the details of the physical address schemes the various technologies use; later chapters will discuss in detail how high level protocols use physical addresses.

2.2 TWO APPROACHES TO NETWORK COMMUNICATION

Whether they provide connections between one computer and another or between terminals and computers, communication networks can be divided into two basic types: Circuit-switched (sometimes called connection oriented) and packet-switched (some times called connectionless). Circuit-switched networks operate by forming a dedicated connection (circuit) between two points. The U.S. telephone system uses circuit-switching technology A telephone call establishes a circuit from the originating phone Through the local switching office, across trunk lines, to a remote switching office, and finally to the destination telephone. While a circuit is in place, the phone equipment samples the microphone repeatedly, encodes the samples digitally, and transmits them across the circuit to the receiver. The sender is guaranteed that the samples can be delivered and reproduced because the circuit provides a guaranteed data path of 64 Kbps (thousand bits per second), the rate needed to send digitized voice. The advantage of circuit switching lies in its guaranteed capacity: once a circuit is established, no other network activity will decrease the capacity of the circuit. One disadvantage of circuit switching is cost: circuit costs are fixed, independent of traffic. For example, one pays a fixed rate for a phone call, even when the two parties do not talk.

Packet-switched networks, the type usually used to connect computers, take an entirely different approach. In a packet-switched network, data to be transferred across a network is divided into small pieces called packets that are multiplexed onto high capacity intermachine connections. A packet, which usually contains only a few hundred bytes of data, carries identification that enables the network hardware to know how to send it to the specified destination. For example, a large file to be transmitted between two machines must be broken into many packets that are sent across the network one at

A time. The network hardware delivers the packets to the specified destination, where software reassembles them into a single file again. The chief advantage of packet-switching is that multiple communications among computers can proceed concurrently, With intermachine connections shared by all pairs of machines that are communicating. The disadvantage, of course, is that as activity increases, a given pair of communicating computers receives less of the network capacity. That is, whenever a packet switched network becomes overloaded, computers using the network must wait before they can send additional packets.

Despite the potential drawback of not being able to guarantee network capacity, packet-switched networks have become extremely popular. The motivations for adopting packet switching are cost and. performance. Because multiple machines can share the network hardware, fewer connections are required and cost is kept low. Because engineers have been able to build high-speed network hardware, capacity is not usually a problem. So many computer interconnections use packet switching that, throughout the remainder of this text, the term network will refer only to packet-switched Networks.

2.3 WIDE AREA AND LOCAL AREA NETWORKS

Packet-switched networks that span large geographical distances (e.g., the continental U.S.) are fundamentally different from those that span short distances (e.g., a single room). To help characterize the differences in capacity and intended use, packet switched technologies are often divided into two broad categories: wide area networks (WANS) and Local Area Networks (LANs). The two categories do not have formal definitions. Instead, vendors apply the terms loosely to help customers distinguish among technologies.

WAN technologies, sometimes called long haul networks, provide communication over large distances. Most WAN technologies do not limit the distance spanned; a WAN can allow the endpoints of a communication to be arbitrarily far apart. For example, a WAN can span a Continent or can join computers across an ocean. Usually, Was operate at slower speeds than LANs, and have much greater delay between connections. Typical speeds for a WAN range from 56 Kbps to 155 MBPS (million bits per second). Delays across a WAN can vary from a few milliseconds to several tenths of a second.

LAN technologies provide the highest speed connections among computers, but sacrifice the ability to span large distances. For example, a typical LAN spans a small area like a single building or a small campus and operates between 10 MBPS and 2 GPS (billion bits per second). Because LAN technologies cover short distances, they

Offer lower delays than WANS. The delay across a LAN can be as short as a few tenths of a millisecond, or as long as 10 milliseconds.

We have already mentioned the general tradeoff between speed and distance: technologies that provide higher speed communication operate over shorter distances. There are other differences among technologies in the categories as well. In LAN technologies, each computer usually contains a network interface device that connects the

Machine directly to the network medium (e.g., a copper wire or coaxial cable). Often, the network itself is passive, depending on electronic devices in the attached computers to generate and receive the necessary electrical signals. In WAN technologies, a net-Work usually consists of a series of complex computers called packet switches interconnected by communication lines and modems. Adding a new switch and another communication line can extend the size of the network. Attaching a user's computer to a WAN means connecting it to one of the packet switches. Each switch along a path in the WAN introduces a delay when it receives a packet and forwards it to the next Switch. Thus, the larger the WAN becomes the longer it takes to route traffic across it.

This chapter discusses software that hides the technological differences between networks and makes interconnection independent of the underlying hardware. To appreciate design choices in the software, it is necessary to understand how it relates to network hardware. The next sections present examples of network technologies that have been used in the Internet, showing some of the differences among them. Later chapters show how the TCP/IP software isolates such differences and makes the communication system independent of the underlying hardware technology.

2.3.1 NETWORK HARDWARE ADDRESSES

Each network hardware technology defines an addressing mechanism that computers use to specify the destination for each packet. Every computer attached to a network is assigned a unique address, usually an integer. A packet sent across a network includes a destination address field that contains the address of the intended recipient.
The destination address appears in the same location in all packets, making it possible for the network hardware to examine the estimation address easily. A sender must know

the address of the intended recipient, and must place the recipient's address in the destination address field of a packet before transmitting the packet.

Each hardware technology specifies how computers are assigned addresses. The hardware specifies, for example, the number of bits in the address as well as the location of the destination address field in a packet. Although some technologies use compatible addressing schemes, many do not. This chapter contains a few examples of hardware addressing schemes; later chapters explain how TCP/IP accommodates diverse hardware addressing schemes.

2.4 ETHERNET TECHNOLOGY

Ethernet is the name given to a popular packet-switched LAN technology invented at Xerox PARC in the early 1970s. Xerox Corporation, Intel Corporation, and Digital Equipment Corporation standardized Ethernet in 1978; IEEE released a compatible version of the standard using the number 802.3. Ethernet has become popular LAN technology; most medium or large corporations use Ethernet. Because Ethernet is so popular, many variants exist; we will discuss the original design first and then cover variants.

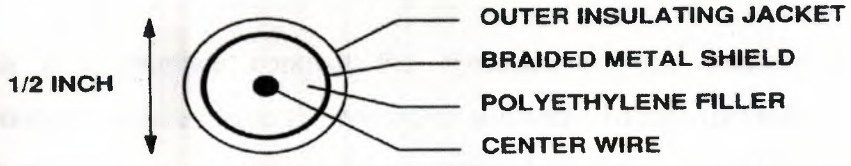


Figure 2.1 A cross-section of the coaxial cable used in the original Ethernet.

Each Ethernet cable is about 1/2 inch in diameter and up to 500 meters long. A resistor is added between the center wire and shield at each end to prevent reflection of electrical signals.

The original Ethernet design used a coaxial cable as Figure 2.1 illustrates. Called the ether, the cable itself is completely passive; all the active electronic components that make the network function are associated with computers that are attached to the network.

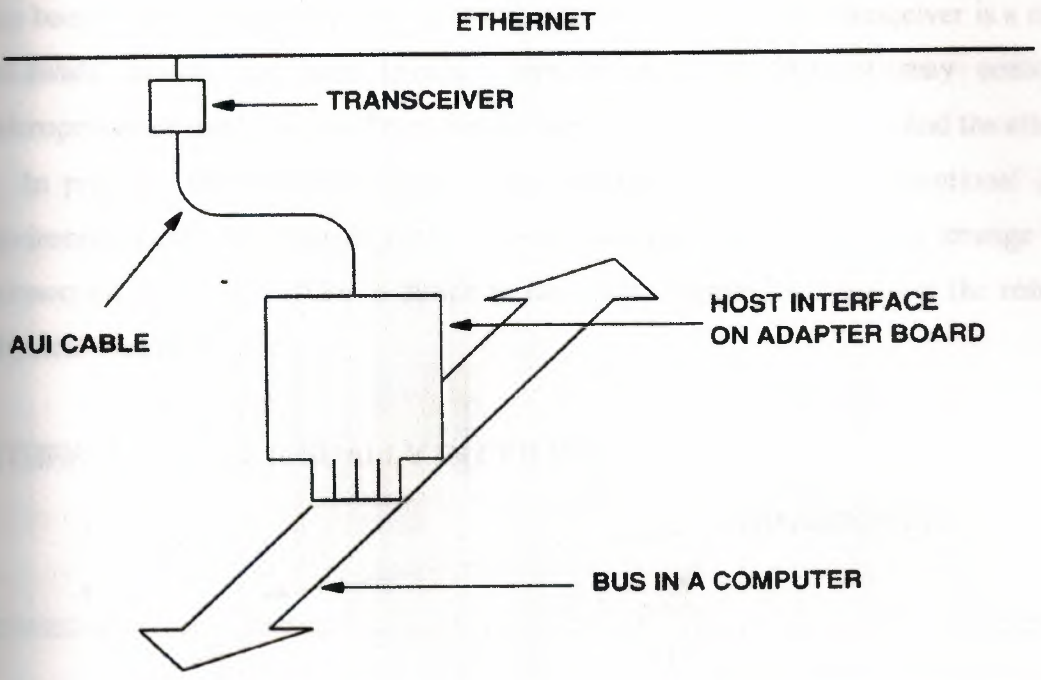


Figure 2.3 the two main electronic components that form a connection between a computer's bus and an Ethernet. The AUI cable that connects the host interface to the transceiver carries power and Signals to control transceiver operation as well as packets being Transmitted or received.

Each host interface controls the operation of one transceiver according to instructions it receives from the computer software. To the operating system software, the interface appears to be an input/output device that accepts basic data transfer instructions

From the computer, controls the transceiver to carry them out, interrupts when the task has been completed, and reports status information. Although the transceiver is a simple hardware device, the host interface can be complex (e.g., it may contain a microprocessor used to control transfers between the computer memory and the ether).

In practice, organizations that use the original Ethernet in a conventional office environment run the Ethernet cable along the ceiling in each hall, and arrange for a connection from each office to attach to the cable. Figure 2.4 illustrates the resulting physical wiring scheme.

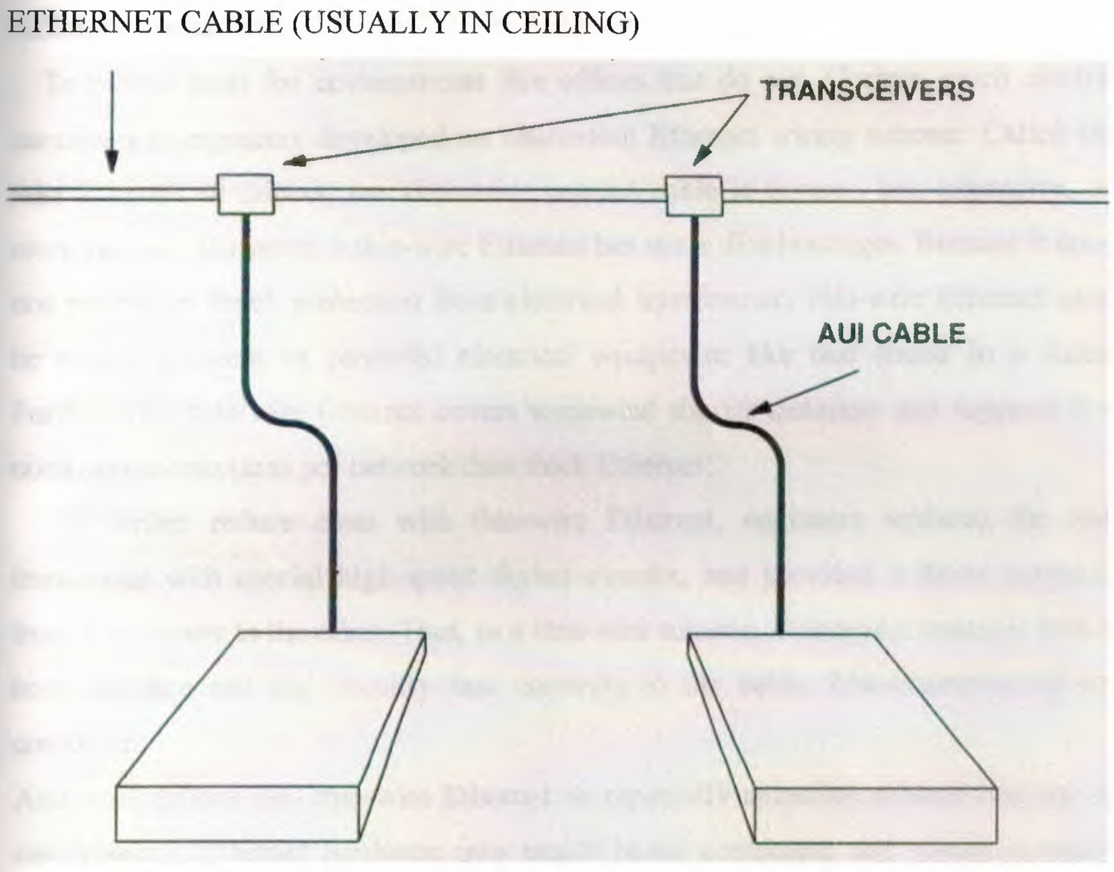


Figure 2.4 the physical connection of two computers to an Ethernet using the original wiring scheme. In an office environment, the Ethernet cable is usually placed in the hallway ceiling; each office has an AUI cable that connects a computer in the office to a transceiver Attached to the Ethernet cable.

2.4.1 THIN-WIRE ETHERNET

Several components of the original Ethernet technology have undesirable properties. For example because a transceiver contains electronic components, it has a nontrivial cost. Furthermore, because transceivers are located with the cable and not with computers, they can be difficult to access or replace. The coaxial cable that fondness the Ether can also be difficult to install. In particular, to provide maximum protection against electrical interference from devices like electric motors, the cable contains heavy shielding that makes it difficult to bend. Finally, an AUI cable is also thick and difficult to bend.

To reduce costs for environments like offices that do not. Contain much electrical interference, engineers developed an alternative Ethernet wiring scheme. Called thin-wire Ethernet or thinnet, the alternative coaxial cable is thinner, less expensive, and, more flexible. However, a thin-wire Ethernet has some disadvantages. Because it does not provide as much protection from electrical interference, thin-wire Ethernet cannot be placed adjacent to powerful electrical equipment like that found in a factory. Furthermore, thin-wire Ethernet covers somewhat shorter distances and supports fewer computer connections per network than thick Ethernet.

To further reduce costs with thin-wire Ethernet, engineers replaced the costly transceiver with special high-speed digital circuits, and provided a direct connection from a computer to the ether. Thus, in a thin-wire scheme, a computer contains both the host interface and the circuitry that connects to the cable. Manufacturers of small computers
And workstations find thin-wire Ethernet an especially attractive scheme because they can integrate Ethernet hardware into single board computers and mount connectors directly on the back of the computer.

Because a thin-wire Ethernet connects directly from one computer to another, the wiring scheme works well when many computers occupy a single room. The thin-wire cable runs directly from one computer to the next. To add a new computer, one only needs to link it into the chain. Figure 2.5 illustrates the connections used with thin-wire Ethernet.

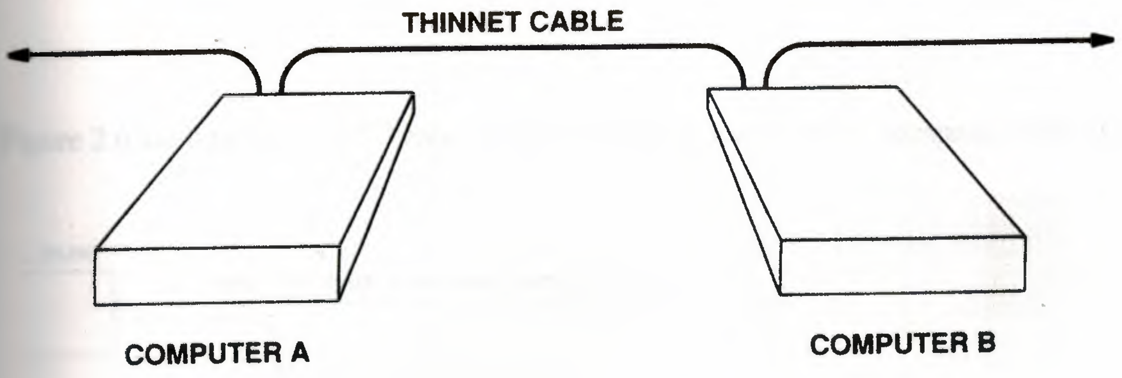


Figure 2.5 the physical connection of two computers using the thinnet-wiring scheme. The ether passes directly from one computer to another; no external transceiver hardware is required.

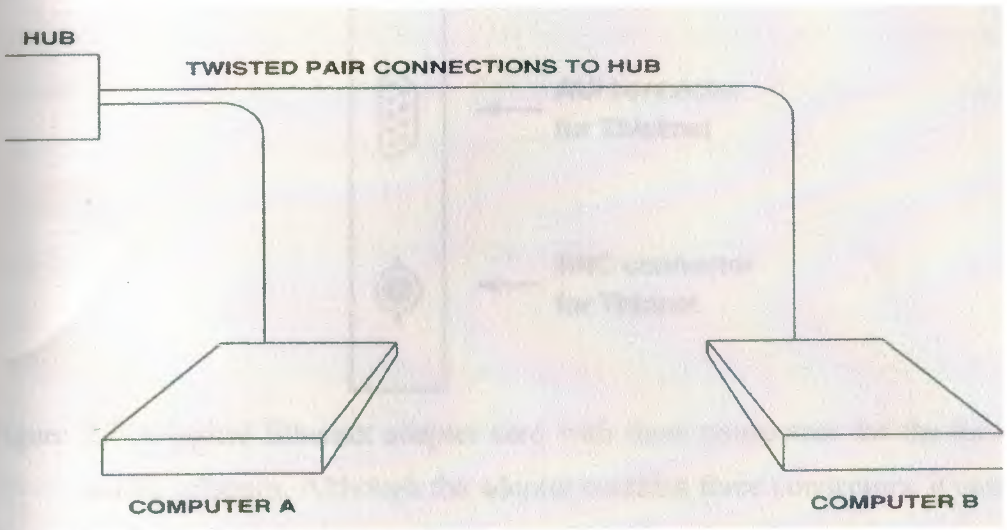
Thin-wire Ethernet are designed to be easy to connect and disconnect. Thin-wire uses BNC connectors, which do not require tools to attach a computer to the cable. Thus, a user can connect a computer to a thin-wire Ethernet without the aid of a technician. Of course, allowing users to manipulate the ether has disadvantages: if a user disconnects the ether, it prevents all machines on the ether from communicating, In many situations, however, the advantages outweigh the disadvantages.

2.4.2 TWISTED PAIR ETHERNET

Advances in technology have made. it possible to build Ethernet that do not need the electrical shielding of a coaxial cable. Called twisted pair Ethernet, the technology allows a computer to access an Ethernet using a pair of conventional unshielded copper wires similar to the wires used to connect telephones. The advantages of using twisted pair wiring are that it further reduces costs and protects other computers on the network from a user who disconnects a single computer. In some cases, a twisted pair technology can make it possible for an organization to use Ethernet over existing telephone wiring without adding new cables.

Known by the technical name 1OBase-T, the twisted pair wiring scheme connects each computer to an Ethernet hub as Figure 2.6 shows.

Figure 2.6 an illustration of Ethernet using twisted pair wiring. Each computer connects



to a hub over a conventional pair of wires.

The hub is an electronic device that simulates the signals on an Ethernet cable. Physically, a hub consists of a small box that usually resides in a wiring closet; a connection between a hub and a computer must be less than 100 meters long. A hub requires power, and can allow authorized personnel to monitor and control its operation over the network. To the host interface in a computer, a connection to a hub appears to operate the same way as a connection to a transceiver. That is, an Ethernet hub provides the same communication capability as a thick or thin Ethernet; hubs merely offer an alternative-wiring scheme.

2.4.3 ADAPTERS AND MULTIPLE WIRING SCHEMES

A connection to thick Ethernet requires an AUI connector, a connection to thin-wire Ethernet requires a BNC connector, and a connection to 1OBase-T requires an RJ45 connector that resembles the modular connectors used with telephones. Many Ethernet products allow each customer to choose a wiring scheme. For example, adapter boards for personal computers often come with three connectors as Figure 2.7 illustrates. Although only one connector can be used at any time, a computer that has such an adapter can's moved from one wiring scheme to another easily.

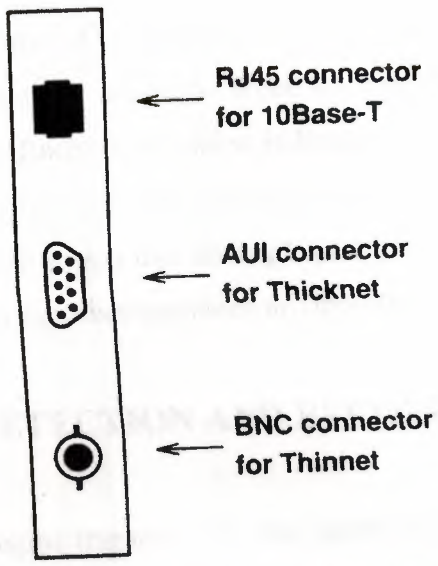

RJ45 connector for 10Base-T
AUI connector for Thicknet
BNC connector for Thinnet

Figure 2.7 A typical Ethernet adapter card with three connectors for the three her net wiring schemes. Although the adapter contains three connectors, it can only use one wiring scheme at any time.

2.4.4 PROPERTIES OF AN ETHERNET

The Ethernet is a 10 MBPS broadcast bus technology with best-effort delivery semantics and distributed access control. It is a bus because. All stations share a single communication channel; it is broadcast because all transceivers receive every transmission. The method used to direct packets from one station to just one other station or a subset of all stations will be discussed later. *For now, it is enough to* understand that transceivers do not distinguish among transmissions - a transceiver passes all packets from the cable to the host interface, which chooses packets the computer should receive and filters out all others. Ethernet is called a best-effort delivery mechanism because the hardware provides no information to the sender about whether the packet was delivered. For example, if the destination machine happens to be powered down, packets sent to it will be lost, and the sender will not be notified. We will see later how the TCP/IP protocols accommodate best-effort delivery hardware.

Ethernet access control is distributed because, unlike some network technologies, Ethernet has no central authority to grant access. The Ethernet access scheme is called Carrier Sense Multiple Access with Collision Detect (CSMAICD). It is CSMA because multiple machines can access the Ethernet simultaneously and each machine determines

Whether the ether is idle by sensing whether a carrier wave is present. When a host interface has a packet to transmit, it listens to the ether to see if a message is being transmitted (i.e., performs carrier sensing). When no transmission is sensed, the host interface starts transmitting. Each transmission is limited in duration (because there is a maximum packet size). Furthermore, the hardware must observe a minimum idle time between transmissions, which means that no single pair of communicating machines can use the network without giving other machines an opportunity for access.

2.4.5 COLLISION DETECTION AND RECOVERY

When a transceiver begins transmission, the signal does not reach all parts of the network simultaneously. Instead it travels along the cable at approximately 80% of the speed of light. Thus, it is possible for two transceivers to both sense that the network is idle and begins transmission simultaneously. When the two electrical signals cross they Become scrambled, such that neither is meaningful. Such incidents are called collisions.

The Ethernet handles collisions in an ingenious fashion. Each transceiver monitors the cable while it is transmitting to see if a foreign signal interferes with its transmission. Technically, the monitoring is called collision detect (CD), making the Ethernet a CSMA/CD network. When a collision is detected, the host interface aborts transmission, waits for activity to subside, and tries again. Care must be taken or the network could wind up with all transceivers busily attempting to transmit and every transmission producing a collision. To help avoid such situations, Ethernet uses a binary exponential bakeoff policy where a sender delays a random time after the first collision, twice as long if a second attempt to transmit also produces a collision, four times as long if a third attempt results in a collision, and so on. The motivation. For exponential bakeoff is that in the unlikely event many stations attempt to transmit simultaneously, a severe
Traffic jam could occur. In such a jam, there is high probability two stations will choose random bakeoffs that are close together. Thus, the probability of another collision is high. By doubling the random delay, the exponential back off strategy quickly spreads the stations' attempts to retransmit over a reasonably long period of time, making the probability of further collisions extremely small.

2.4.6 ETHERNET CAPACITY

The standard Ethernet is rated at 10 MBPS, which means that data can be transmitted onto the cable at 10 million bits per second. Although a computer can generate data at Ethernet speed, raw network speed should not be thought of as the rate at which two computers can exchange data. Instead, network speed should be thought of as a measure of network total traffic capacity. Think of a network as a highway connecting multiple cities, and think of packets as cars on the highway. High bandwidth makes it possible to carry heavy traffic loads, while low bandwidth means the highway cannot carry as much traffic. A 10 MBPS Ethernet, for example, can handle a few computers that generate heavy loads, or many computers that generate light loads.

CHAPTER 3
THE DOMAIN NAME SYSTEM

3.1 INTRODUCTION

Before 1980, the ARPANET had only a few hundred networked computers. The computer name-to-address mapping was contained in a single file called Hosts.txt. This file was stored on the host computer of the Stanford Research Institute's Network Information Center (SRI-NIC) in Menlo Park, California. Other host computers on the ARPANET copied the Hosts.txt file from the SRI-NIC to their sites as needed.

3.2 NAMES FOR MACHINES

The earliest computer systems forced users to understand numeric addresses for objects like system tables and peripheral devices. Timesharing systems advanced computing by allowing users to invent meaningful symbolic names for both physical objects (e.g., peripheral devices) and abstract objects (e.g., files). A similar pattern has emerged in computer networking. Early systems supported point-to-point connections between computers and used low-level hardware addresses to specify machines. Internetworking introduced universal addressing as well as protocol software to map universal address into low-level hardware addresses. Because most computing environments contain multiple machines, users need meaningful, symbolic names to identify them.

Early machine names reflected the small environment in which they were chosen It was quite common for a site with a handful of machines to choose names base the machines' purposes. For example, machines often had names like research, production, accounting, and development. Users find such names preferable to cumber hardware addresses.

Although the distinction between address and name is intuitively appealing, artificial. Any name is merely an identifier that consists of a sequence of chars chosen from a finite alphabet. Names are only useful if the system can efficiently them to the object they denote. Thus, we think of an IP address as a low-level r, and we say that users prefer

High-level names for machines.

The form of high-level names is important because it determines how name translated to lower-level names or bound to objects, as well as how name assignments are authorized. When only a few machines interconnect, choosing names is easy any form will suffice. On the Internet, to which over four million machines connect choosing symbolic names becomes difficult. For example, when its main departmental computer was connected to the Internet in 1980, the Computer Science Department at Purdue University chose the name purdue to identify the connected machine. The list of potential conflicts contained only a few dozen names. By mid 1986, the official of hosts on the Internet contained 3100 officially registered names and 6500 Official aliases. Although the list was growing rapidly in the 1980s, most sites had additional machines (e.g., personal computers) that were not registered.

3.3 FLAT NAMESPACE

The original set of machine names used throughout the Internet formed namespace in which each name consisted of a sequence of characters without any further structure. In the original scheme, a central site, the Network Information (NIC), administered the namespace and determined whether a new name was app ate (i.e., it prohibited obscene names or new names that conflicted with existing names Later, the NIC was replaced by the INTERnet Network Information Center (INTR).

The chief advantage of a flat namespace is that names are convenient and short; the chief disadvantage is that a flat namespace cannot generalize to large sets of machines for both technical and administrative reasons. First, because names are drawn from a single set of identifiers, the potential for conflict increases as the number of site increases. Second, because authority for adding new names must rest at a single site, the administrative workload at that central site also increases with the number of sites. To understand the severity of the problem, imagine a rapidly growing Internet with thousands of sites, each of which has hundreds of individual personal computers and workstations. Every time someone acquires and connects a new personal computer, the central authority must approve its name. Third, because the name-to-address bindings change frequently, the cost of maintaining correct copies of the entire list at each site is high and increases as the number of sites increases. Alternatively, if the name database resides at a single site, network traffic to that site increases with the number of sites.

3.4 Hierarchical Names

How can a naming system accommodate large, rapidly expanding set of names without requiring a central site to administer it? The answer lies in decentralizing the naming mechanism by delegating authority for parts of the namespace and distributing responsibility for the mapping between names and addresses. TCP/IP Internets use such a scheme. Before examining the details of the TCP/IP scheme, we will consider the motivation and intuition behind it.

The partitioning of a namespace must be defined in a way that supports efficient name mapping and guarantees autonomous control of name assignment. Optimizing only for efficient mapping can lead to solutions that retain a flat namespace and reduce traffic by dividing the names among multiple mapping machines. Optimizing only for administrative ease can lead to solutions that make delegation of authority easy but name mapping expensive or complex.

To understand how the namespace should be divided, consider the internal structure of large organizations. At the top, a chief executive has overall responsibility. Because the chief executive cannot oversee everything, the organization may be partitioned into divisions, with an executive in charge of each division. The chief executive grants each division autonomy within specified limits. More to the point, the executive in charge of a particular division can hire or fire employees, assign offices, and delegate authority, without obtaining direct permission from the chief executive.

Besides making it easy to delegate authority, the hierarchy of a large organization introduces autonomous operation. For example, when an office worker needs information like the telephone number of a new employee, he or she begins by asking local clerical workers (who may contact clerical workers in other divisions). The point is that although authority always passes down the corporate hierarchy, information can flow across the hierarchy from one office to another.

3.5 DELEGATION OF AUTHORITY FOR NAMES

A hierarchical naming scheme works like the management of a large organization. The namespace is partitioned at the top level, and authority for names in subdivisions is passed to designated agents. For example, one might choose to partition the namespace based on site name and to delegate to each site responsibility for maintaining names within its partition. The topmost level of the hierarchy divides the namespace and delegates authority for each division; it need not be bothered by changes within a division.

The syntax of hierarchically assigned names often reflects the hierarchical delegation of authority used to assign them. As an example, consider a namespace with names of the form:

Local.site

Where site is the site name authorized by the central authority, local is the part of a name controlled by the site, and the period (".") is a delimiter used to separate them. When the topmost authority approves adding a new site, X, it adds X to the list of valid sites and delegates to site X authority for all names that end in ". X ".

3.6 SUBSET AUTHORITY

In a hierarchical namespace, authority may be further subdivided at each level. In our example of partition by sites, the site itself may consist of several administrative groups, and the site authority may choose to subdivide its namespace among the groups. The idea is to keep subdividing the namespace until each subdivision is small enough to be manageable.

Syntactically, subdividing the namespace introduces another partition of the name. For example, adding group subdivision to names already partitioned by site produces the following name syntax: Local.group. site

Because the topmost level delegate's authority, group names do not have to agree among all sites. A university site might choose group names like engineering, science, and arts, while a corporate site might choose group names like production, accounting, and personnel.

The U 'S. telephone system provides another example of a hierarchical naming syntax. The 10 digits of a phone number have been partitioned into a 3-digit area code, 3-digit exchange, and 4-digit subscriber number within the exchange. Each exchange has authority for assigning subscriber numbers within its piece of the namespace. Although it is possible to group arbitrary subscribers into exchanges and to group arbitrary exchanges into area codes, the assignment of telephone numbers is not capricious; they are carefully chosen to make it easy to route phone calls across the telephone network.

Local. group. site

The domain names, the period delimiter is pronounced "dot.

The telephone example is important because it illustrates a key distinction between hierarchical naming scheme used in a TCP/IP internet and other hierarchies: partitioning the set of machines owned by an organization along lines of authority does not necessarily imply partitioning by physical location. For example, it could be that at university, a single building houses the mathematics department as well as the computer science department. It might even turn out that although the machines from se two groups fall under completely separate administrative domains, they connect to same physical network. It also may happen that a single group owns machines on several physical networks. For these reasons, the TCP/IP naming scheme allows arbitrary delegation of authority for the hierarchical namespace without regard to physical connections. The concept can be summarized:

In a TCP/IP Internet, hierarchical machine names are assigned according to the structure of organizations that obtain authority for parts of the namespace, not necessarily according to the structure of the physical network interconnections.

Of course, at many sites the organizational hierarchy corresponds with the structure of physical network interconnections. At a large university, for example, most departments have their own local area network. If the department is assigned part of the naming hierarchy, all machines that have names in its part of the hierarchy will also connect to a single physical network.

3.7 TCP/IP INTERNET DOMAIN NAMES

The mechanism that implements a machine name hierarchy for TCP/IP Internets is called the Domain Name System (DNS). DNS has two, conceptually independent aspects, The first is abstract: it specifies the name syntax and rules for delegating authority over names. The second is concrete: it specifies the implementation of a distributed computing system that efficiently maps names to addresses. This section considers the name syntax, and later sections examine the implementation.

The domain name system uses a hierarchical naming scheme known as domain names. As in our earlier examples, a domain name consists of a sequence of subnames separated by a delimiter character, the period. In our examples we said that individual ions of the name might represent sites or groups, but the domain system simply each section a label. Thus, the domain name

cs.purdue.edu

Contains three labels: cs, purdue, and edu. Any suffix of a label in a domain name is called a domain. In the above example the lowest level domain is cs. purdue. Edu. Domain name for the Computer Science Department at Purdue University), the id level domain is purdue.edu (the domain name for Purdue University), and the cs.purdue.edu top-level domain is edu (the domain name for educational institutions). As the example shows, domain names are written with the local label first and the top domain last. As we will see, writing them in this order makes it possible to compress messages that contain multiple domain names.

3.8 OFFICIAL AND UNOFFICIAL INTERNET DOMAIN NAMES

In theory, the domain name standard specifies an abstract hierarchical namespace with arbitrary values for labels. Because the domain system dictates only the form of names and not their actual values, it is possible for any group that builds an instance of the domain system to choose labels for all parts of its hierarchy. For example, a private company can establish a domain hierarchy in which the top-level labels specifies corporate subsidiaries, the next level labels specify corporate divisions, and the lowest level labels specify departments.

However, most users of the domain technology follow the hierarchical labels used by the official Internet domain system. There are two reasons. First, as we will see, the Internet scheme is both comprehensive and flexible. It can accommodate a wide variety of organizations, and allows each group to choose between geographical or organizational naming hierarchies. Second, most sites follow the Internet scheme so they can attach their TCP/IP installations to the global Internet without changing names. Because the Internet naming scheme dominates almost all uses of the domain name system, examples throughout the remainder of this chapter have labels taken from the Internet naming hierarchy. Readers should remember that, although they are most likely to en- counter these particular labels, the domain name system technology can be used with other labels if desired. The Internet authority has chosen to partition its top level into the domains listed in Figure 3.1.

Domain Name	meaning
Com	Commercial organizations
EDU	Educational institutions
GOV	Government institutions
MIL	Military groups
NET	Major network support centers
ORG	Organizations other than those above
ARPA	Temporary ARPANET
INT	International organizations
COUNTRY CODE	Each country (geographic scheme)

Figure 3.1 the top-level Internet domains and their meanings. Although labels are shown in upper case. Domain name system comparisons

Conceptually, the top-level names permit two completely different naming hierarchies: geographic and organizational. The geographic scheme divides the universe of machines by country. Machines in the United States fall under the top-level domain US; when a foreign country wants to register machines in the domain name system, the central authority assigns the country a new top-level domain with the country's international standard 2-letter identifier as its label. The authority for the US domain has chosen to divide it into one second-level domain per state. For example, the domain for the state of Virginia is

Va.us

As an alternative to the geographic hierarchy, the top-level domains also allow organizations to be grouped by organizational type. When an organization wants to participate in the domain naming system, it chooses how it wishes to be registered and requests approval. The central authority reviews the application and assigns the organization a subdomain under one of the existing top-level domains. For example, it is possible for a university to register itself as a second-level domain under EDU (the usual practice), or to register itself under the state and country in which it is located. So far, few organizations have chosen the geographic hierarchy; most prefer to register under COM, EDU, MIL, or GOV. There are two reasons. First, geographic names are longer and therefore more difficult to type. Second, geographic names are much more difficult to discover or guess. For example, Purdue University is located in West Lafayette, Indiana. While a user could easily guess an organizational name, like purdue.edu, a geographic name is often difficult to guess because it is usually an abbreviation, like

laf.in. us.

Another example may help clarify the relationship between the naming hierarchy and authority for names. A machine named xinu in the Computer Science Department at Purdue University has the official domain name

xinu.cs.purdue.edu

The machine name was approved and registered by the local network manager in the Computer Science Department. The department manager had previously obtained authority for the subdomain cs.purdue. edu from a university network authority, who had obtained permission to manage the subdomain purdue. edu from the Internet authority.

The Internet authority retains control of the edu domain, so new universities can only be added with its permission. Similarly, the university network manager at Purdue University retains authority for the purdue. edu subdomain, so new third-level domains may only be added with the manager's permission.

Figure 3.2 illustrates a small part of the Internet domain name hierarchy. As the figure shows, Digital Equipment Corporation, a commercial organization, registered as dec.com, Purdue University registered as purdue.edu, and the National Science Foundation, a government agency registered as nsf.gov in corporation for

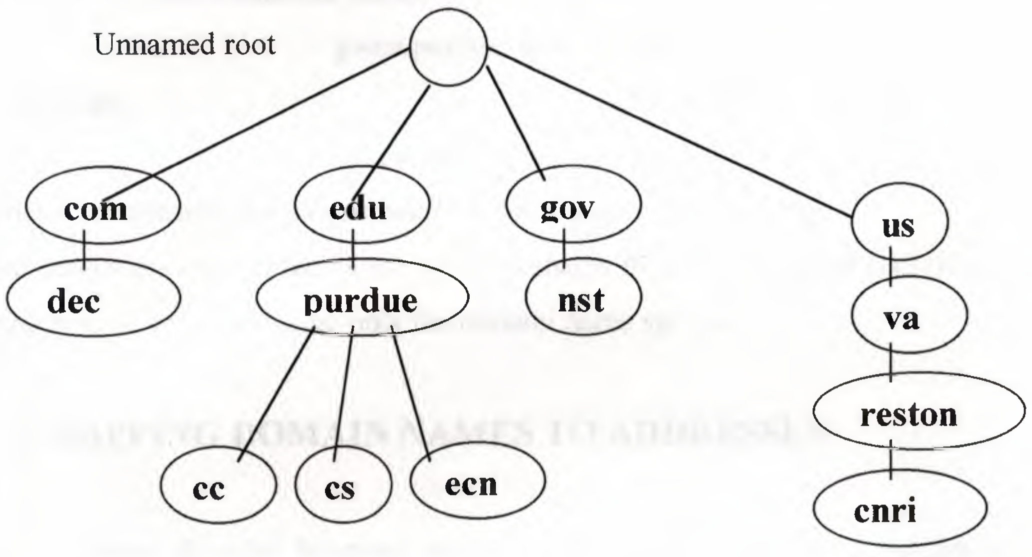


Figure 3.2 A small part of the Internet domain name hierarchy (tree). In practice, the tree is broad and flat-, most host entries appear by The fifth level.

3.9 ITEMS NAMED AND SYNTAX OF NAMES

The domain name system is quite general because it allows multiple naming hierarchies to be embedded in one system. To allow clients to distinguish among multiple kinds of entries, each named item stored in the system is assigned a type that specifies whether it is the address of a machine, a mailbox, a user, and so on. When a client asks the domain system to resolve a name, it must specify the type of answer

red. For example, when an electronic mail application uses the domain system to
lve a name, it specifies that the answer should be the address of a mail exchanger. A
ote login application specifies that it seeks a machine's IP address. It is important to
erstand the following:

given name may map to more than one item in the domain system. The client
cifies the type of object desired when resolving a name, and the server returns
ects of that type.

The syntax of a name does not determine what type of object it names or the class of
tocol suite. In particular, the number of labels in a name does not determine whether
name refers to an individual object (machine) or a domain. Thus, in our example, it
possible to have a machine named

gwen.purdue. edu

en though

cs.pu.rdue.edu

ames a subdomain. We can summarize this important point:

One cannot distinguish the names of subdomains from the names of individual objects
or the type of an object using only the domain name syntax.

3.10 MAPPING DOMAIN NAMES TO ADDRESSES

In addition to the rules for name syntax and delegation of authority, the domain name scheme includes an efficient, reliable, general purpose, distributed system for mapping names to addresses. The system is distributed in the technical sense, meaning that a set of servers operating at multiple sites cooperatively solve the mapping problem. It is efficient in the sense that most names can be mapped locally; only a few re- quire Internet traffic. It is general purpose because it is not restricted to machine names (although we will use that example for now). Finally, it is reliable in that no single machine failure will prevent the system from operating correctly.

The domain mechanism for mapping names to addresses consists of independent, cooperative systems called name servers. A name server is a server program that supplies name-to-address translation, mapping from domain names to IP addresses. Often, server software executes on a dedicated processor, and the machine itself is called the name server. the client software, called a name resolver, uses one or more name servers when translating a name.

The easiest way to understand how domain servers work is to imagine them arranged in a tree structure that corresponds to the naming hierarchy, as Figure 3.3 illustrates. The root of the tree is a server that recognizes the top-level domains and knows which server resolves each domain. Given a name to resolve, the root can choose the correct server for the name.

Correct server for that name. At the next level, a set of name servers each provides answers for one top-level domain (e.g., edu). A server at this level knows which servers can resolve each of the subdomains under its domain. At the third level of the tree, name servers provide answers for subdomains (e.g., purdue under edu). The con- Links in the conceptual tree do not indicate physical network connections. Instead, they show which other name servers a given server knows and contacts. The servers themselves may be located at arbitrary locations on an Internet. Thus the tree of servers is an abstraction that uses an Internet for communication.

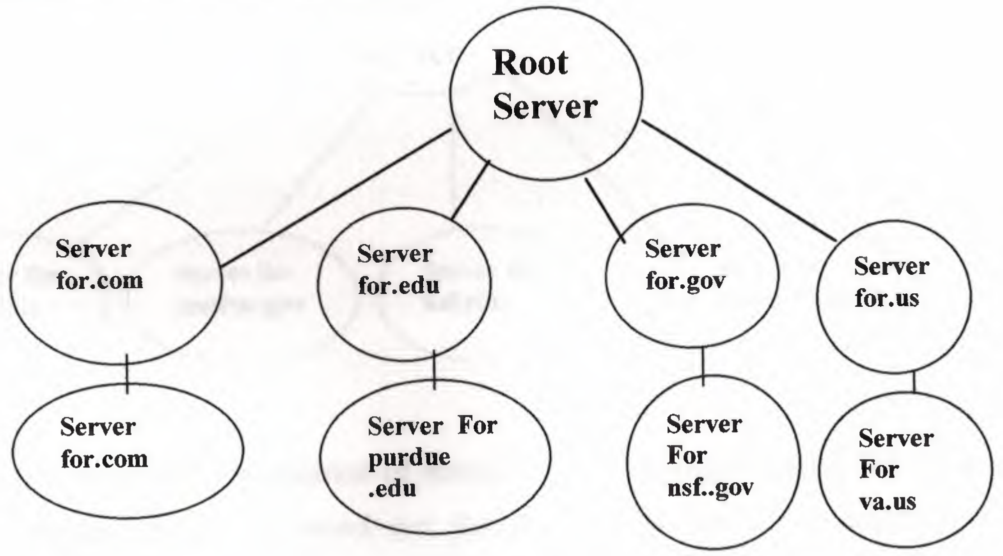


Figure 3.3 the conceptual arrangement of domain name servers in a tree that corresponds to the naming hierarchy, In theory, each server Knows the addresses of all lower-level servers for all sub- domains Within the domain it handles.

If servers in the domain system worked exactly as our simplistic model suggests the relationship between connectivity and authorization would be quite simple. When authority was granted for a subdomain, the organization requesting it would need to establish a domain name server for that subdomain and link it into the tree.

In practice, the relationship between the naming hierarchy and the tree of servers is not as simple as our model implies. The tree of servers has few levels because a single physical server can contain all of the information for large parts of the naming hierarchy. In particular, organizations often collect information from all of their subdomains into a single server. Figure 3.4 shows a more realistic organization of servers for the naming hierarchy of Figure3.2.

A root server contains information about the root and top-level domains. And each the root server for domain purdue.edu (i.e., the root server knows which server handles purdue.edu, and the entire domain information for purdue resides server).

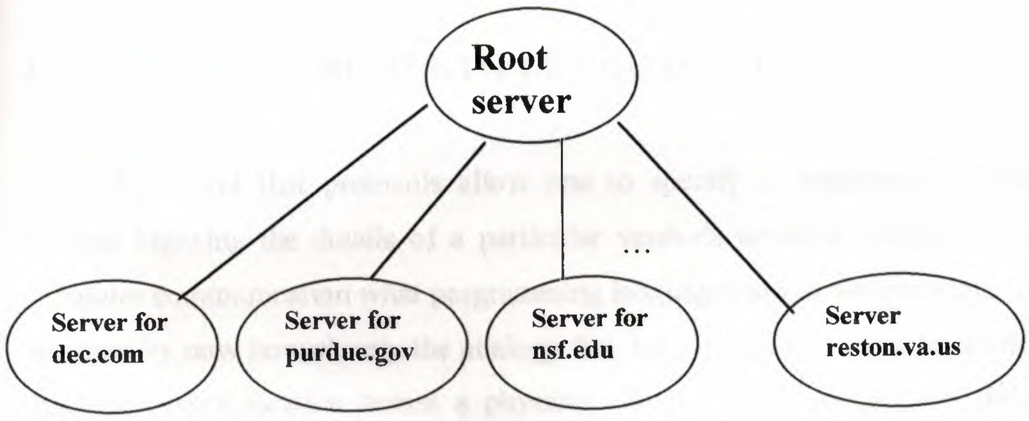


Figure 3.4 a realistic organization of servers for the naming hierarchy of figure 3.2. Because the tree is board and flat, few servers need to be contacted when resolving a name

CHAPTER 4
PROTOCOL LAYERING AND INTERNET PROTOCOL

4.1 INTRODUCTION

Previous chapter2 review the architectural foundations of internetworking describe how hosts and routers forward Internet datagrams and present mechanisms used to map IP addresses to physical network addresses. This chapter considers the structure of the software found in hosts and routers that carries out network communication. It presents the general principle of layering, shows how layering makes Internet Protocol software easier to understand and build, and traces the path of datagrams through the protocol software they encounter when traversing a TCP/IP Internet.

4.2 THE NEED FOR MULTIPLE PROTOCOLS

We have said that protocols allow one to specify or understand communication without knowing the details of a particular vendor's network hardware. They are to computer communication what programming languages are to computation. It should be apparent by now how closely the analogy fits. Like assembly language, some protocols describe communication across a physical network. For example, the details of the Ethernet frame format, network access policy, and frame error handling comprise a protocol that describes communication on an Ethernet. Similarly, the details of IP addresses, the datagram format, and the concept of unreliable, connectionless delivery comprise the Internet Protocol.

Complex data communication systems do not use a single protocol to handle all Hardware failure. A host or router may fail either because the hardware fails or because the operating system crashes. A network transmission link may fail or accidentally be disconnected. The protocol software needs to detect such failures and recover from them if possible.

- Network congestion. Even when all hardware and software operates correctly, networks have finite capacity that can be exceeded. The protocol software needs to arrange ways that a congested machine can suppress further traffic.

- Packet delay or loss. Sometimes, packets experience extremely long delays or are lost. The protocol software needs to learn about failures or adapt to long delays.
- Data corruption. Electrical or magnetic interference or hardware failures can cause transmission errors -that corrupt the contents of transmitted data. Protocol software needs to detect and recover from such errors.
- A Data duplication or sequence errors. Networks that offer multiple routes may deliver data out of sequence or may deliver duplicates of packets. The protocol software needs to reorder packets and remove any duplicates.

Taken together, all these problems seem overwhelming. It is difficult to understand how to write a single protocol that will handle them all. From the analogy with programming languages, we can see how to conquer the complexity. Program translation has been partitioned into four conceptual subproblems identified with the software that handle each subproblem: compiler, assembler, link editor, and loader. The division makes it possible for the designer to concentrate on one subproblem at a time, and for the implementor to build and test each piece of software independently. We will see that protocol software is partitioned similarly.

Two final observations about our programming language analogy will help clarify the organization of protocols. First, it should be clear that pieces of translation software must agree on the exact format of data passed between them. For example, the data passed from the compiler to the assembler consists of a program defined by the assembly programming language. Thus, we see how the translation process involves multiple programming languages. The analogy will hold for communication software, where we will see that multiple protocols define the interfaces between the modules of communication software. Second, the four parts of the translator form a linear sequence in which output from the compiler becomes input to the assembler, and so on. Protocol software also uses a linear sequence.

4.3 THE CONCEPTUAL LAYERS OF PROTOCOL SOFTWARE

Think of the modules of protocol software on each machine as being stacked

Vertically into layers, as in Figure 4.1. Each layer takes responsibility for handling one part of the problem.

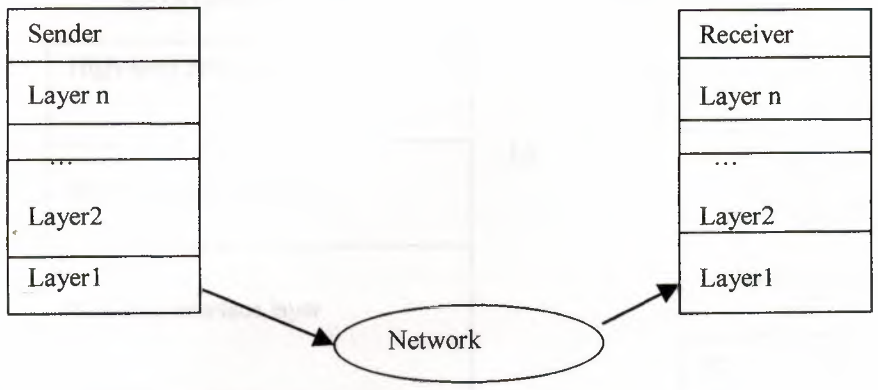


Figure 4.1 the conceptual organization of protocol software in layers.

Conceptually, sending a message from an application program on one machine to an application program on another means transferring the message down through successive layers of protocol software on the sender's machine, transferring the message across the network, and transferring the message up through successive layers of protocol software on the receiver's machine.

In practice, the protocol software is much more complex than the simple model of Figure 4. I indicate. Each layer makes decisions about the correctness of the message and chooses an appropriate action based on the message type or destination address.

For example, one layer on the receiving machine must decide whether to keep the message or forward it to another machine. Another layer must decide which application program should receive the message.

To understand the difference between the conceptual organization of protocol software and the implementation details, consider the comparison shown in Figure 4.2. The conceptual diagram in Figure 4.2a shows an Internet layer between a high-level protocol layer and a network interface layer. The realistic diagram in Figure 4.2b shows that the IP software may communicate with multiple high-level protocol modules and with multiple network interfaces.

Although a diagram of conceptual protocol layering does not show all details, it does help explain the general ideas. For example, Figure 11.3 shows the layers of protocol software used by a message that traverses three networks. The diagram shows only the network interface and Internet Protocol layers in routers because only those layers are needed to receive, route, and then send datagrams. We understand that any machine attached to two networks must have two network interface modules, even

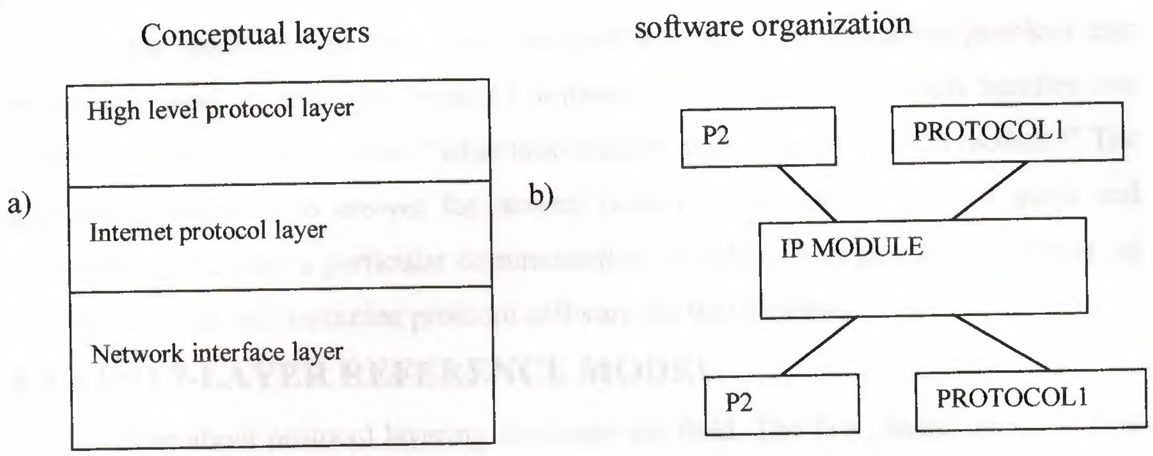


Figure 4.2 A comparison of (a) conceptual protocol layering and (b) a realistic view of software organization showing multiple network Interfaces below IP and multiple protocols above it.

As Figure 4.3 shows, a sender on the original machine transmits a message, which the IP layer places in a datagram, and sends across network 1. On intermediate machines the datagram passes up to the IP layer which routes it back out again (on a different network). Only when it reaches the final destination machine does IP extract the Message and pass it up to higher layers of protocol software.

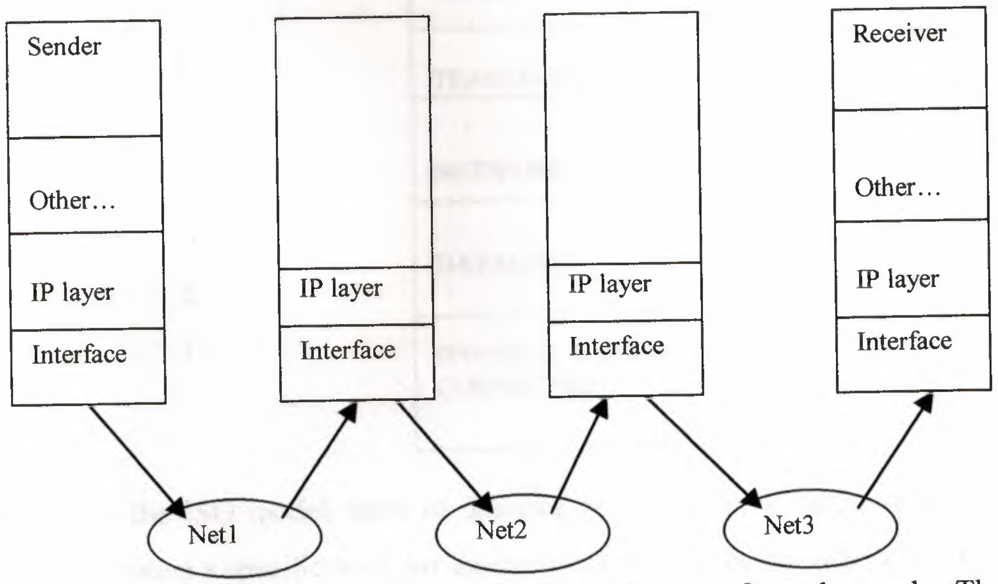


Figure 4.3 the path of a message traversing the Internet from the sender Through two intermediate machines to the receiver, intermediate Machines only send the datagram to the IP software layer.

4.4 FUNCTIONALITY OF THE LAYERS

Once the decision has been made to partition the communication problem into subproblem and organize the protocol software into modules that each handles one subproblem, the question arises: "what functionality should reside in each module?" The question is not easy to answer for several reasons. First, given a set of goals and constraints governing a particular communication problem, it is possible to choose an organization that will optimize protocol software for that problem.

4.4.1 ISO 7-LAYER REFERENCE MODEL

Two ideas about protocol layering dominate the field. The first, based on work done by the International Organization for Standardization (ISO), is known as ISO's Reference Model of Open
System Interconnection often referred to as the ISO model.
The ISO model contains 7 conceptual layers organized as Figure 4.4 shows.

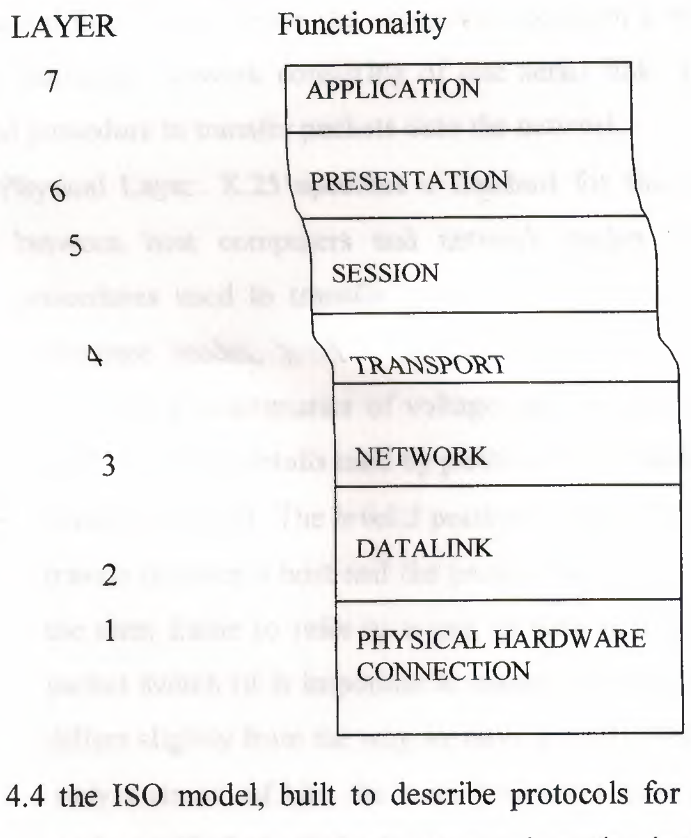


Figure 4.4 the ISO model, built to describe protocols for a single network, does not contain a specific level for internetwork routing in the same way TCP protocols do.

unreliable, the level 2 protocol specifies an exchange of acknowledgements that allows the two machines to know when a frame has been transferred successfully.

One commonly used level 2 protocol, named the High Level Data Link Communication, and is best known by its acronym, HDLC. Several versions of HDLC exist, with the most recent known as HDLC/LAPB. It is important to remember that successful transfer at level 2 means a frame has been passed to the network packet switch for delivery; it does not guarantee that the packet switch accepted the packet was able to.

- Network Layer. The ISO reference model specifies that the third level contains functionality that completes the definition of the interaction between host and network. Called the network or communication subnet layer. This level defines the basic unit of Transfer across the network and includes the concepts of Destination addressing and routing. Remember that in the X.25 World, communication between host and packet switch is conceptually isolated from the traffic that is being passed. Thus, the network might allow packets defined by level 3 protocols to be larger than the size of frames that can form the network expects and uses level 2 to transfer it (possibly in pieces) to the packet switch. Level 3 must also respond to network congestion problems.
- Transport Layer. Level 4 provides end-to-end reliability by having the destination host communicate with the source host. The idea here is that even though lower layers of protocols provide reliable checks at each transfer, the end-to-end layer double checks to make sure that no machine in the middle failed.
- Session Layer. Higher levels of the ISO model describe how protocol software can be organized to handle all the functionality needed by application programs. The ISO committee considered the problem of remote terminal access so fundamental that they assigned layer 5 to handle it. In fact, the central service offered by early public data networks consisted of terminal to host interconnection. The carrier provides a special purpose host

computer called a Packet Assembler And Dissembler (PAD) on the network with dialup access. Subscribers, often travelers who carry their own computer and modem, dial up the local PAD, make a network connection to the host with which they wish to communicate, and log in. Many carriers choose to make using the network for long distance communication less expensive than direct dialup.

- Presentation Layer. ISO layer 6 is intended to include functions that many application programs need when using the network. Typical examples include standard routines that compress text or convert graphics images into bit streams for transmission across a network. For example an ISO standard known as Abstract Syntax Notation I (ASN.1), provides a representation of data that application programs use. One of the TCP/IP protocols, SNMP, also uses ASN.1 to represent data.
- Application Layer. Finally, ISO layer 7 includes application programs that use the network. Examples include electronic mail or file transfer programs. In particular, the ITU-TS has devised a protocol for electronic mail known as the X.400 standard. In fact, the ITU and ISO worked jointly on message handling systems; the ISO version is called MOTIS.

4.5.1 THE TCP/IP INTERNET LAYERING MODEL

The second major layering model did not arise from a standards committee, but came instead from research that led to the TCP/EP protocol suite. With a little work, the ISO model can be stretched to describe the TCP/IP layering scheme, but the underlying Broadly speaking, TCP/IP software is organized into four conceptual layers that build on a fifth layer of hardware. Figure 4.5 shows the conceptual layers as well as the form of data as it passes between them.

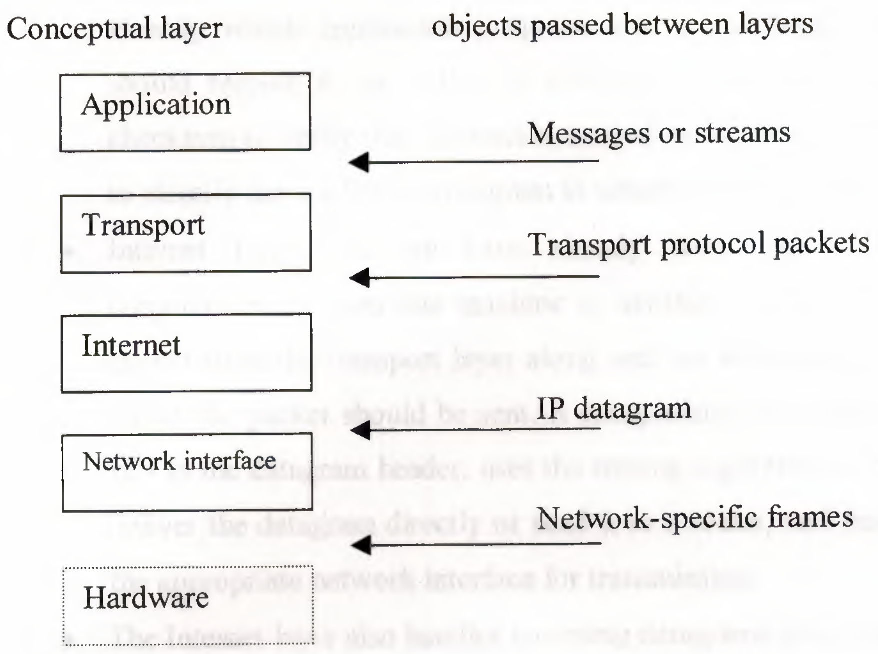


Figure 4.5 the 4 conceptual layers of TCP/IP software and the form of Objects passed between layers. The layer labeled network interface is sometimes called the data link layer.

- Application Layer. At the highest level, users invoke application programs that access services available 'across a TCP/IP Internet. An application interacts with one of the transport level protocols to send or receive data. Each application program chooses the style of transport needed, which can be either a sequence of individual messages or a continuous stream of bytes. The application program passes data in the required form to the transport level for delivery.
- Transport Layer. The primary duty of the transport layer is to provide communication from one application program to another. Such communication is often called end-to-end. The transport layer may regulate flow of information. It may also provide reliable transport, ensuring that data arrives without error and in sequence. To do so, transport protocol software arranges to have the receiving side send back acknowledgements and the sending side retransmit lost packets. The transport software divides the stream of data being transmitted into small pieces (sometimes called packets) and passes each packet along with a destination address to the next layer for transmission .to the next lower layer. To do so, it adds additional information to each packet, including codes that

identify which application program sent it and which application program should receive it, as well as a checksum. The receiving machine uses the checksum to verify that the packet arrived intact, and uses the destination code to identify the application program to which it should be delivered.

- Internet Layer. As we have already seen, the Internet layer handles communication from one machine to another. It accepts a request to send a packet from the transport layer along with an identification of the machine to which the packet should be sent. It encapsulates the packet in an IP datagram, fills in the datagram header, uses the routing algorithm to determine whether to deliver the datagram directly or send it to a router, and passes the datagram to the appropriate network interface for transmission.
- The Internet layer also handles incoming datagrams, checking their validity, and uses the routing algorithm to decide whether the datagram should be processed locally or for-warded. For datagrams addressed to the local machine, software in the internet layer deletes the datagram header, and chooses from among several transport protocols the one that will handle the packet. Finally, the Internet layer sends ICMP error and control messages as needed and handles all incoming ICNLP messages.
- Network Interface Layer, The lowest level TCP/IP software comprises a network interface layer, responsible for accepting IP datagrams and transmitting them over a specific network. A network interface may consist of a device driver (e.g., when the network is a local area network to which the machine attaches directly) or a complex subsystem that uses its own data link protocol (e.g., when the network consists of packet switches that communicate with hosts using HDLC).

4.6 DIFFERENCES BETWEEN X.25 AND INTERNET LAYERING

There are two subtle and important differences between the TCP/IP layering scheme and the X.25 scheme. The first difference revolves around the focus of attention on reliability, while the second involves the location of intelligence in the overall system.

4.7 THE PROTOCOL LAYERING PRINCIPLE

Independent of the particular layering scheme used, or the function of the layers the operation of layered protocols is based on a fundamental idea. The idea, called the layering principle, can be summarized succinctly:

Layered protocols are designed so that layers n at the destination
Receives exactly the same object sent by layer n at the source.

The layering principle explains why layering is such a powerful idea. It allows the protocol designer to focus attention on one layer at a time, without worrying about how lower layers perform. For example, when building a file transfer application, the designer thinks only of two copies of the application program executing on two machines and concentrates on the messages they need to exchange for file transfer. The designer assumes that the application on one host receives exactly what the application on the other host sends.

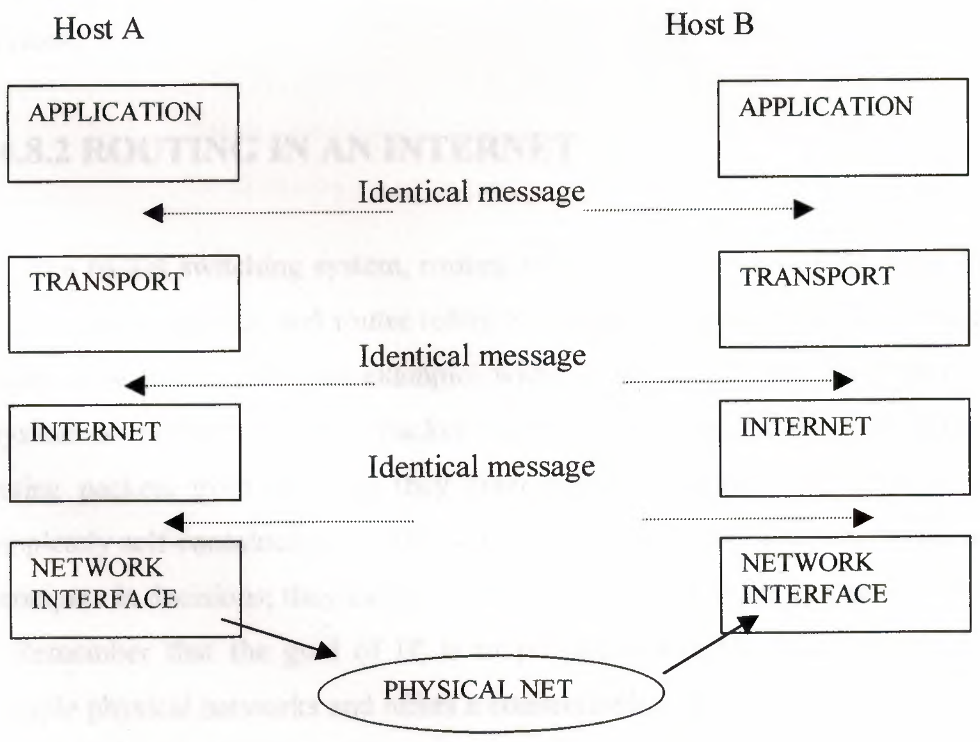


Figure 4.6

4 .8 INTERNET PROTOCOL: ROUTING IP DATAGRAMS

4.8.1 INTRODUCTION

We have seen that all Internet services use an underlying, connectionless packet delivery system, and that the basic unit of transfer in a TCP/IP Internet is the IP datagram. This chapter adds to the description of connectionless service by describing how routers forward IP datagrams and deliver them to their final destinations. We think of the datagram format from Chapter 7 as characterizing the static aspects of the Inter net Protocol. The description of routing in this chapter characterizes the operational aspects. The next chapter concludes our presentation of IP by describing how errors are handled; later chapters show how other protocols use IP to provide higher-level services.

4.8.2 ROUTING IN AN INTERNET

In a packet switching system, routing refers to the process of choosing a path over which to send packets, and router refers to a computer making such a choice. Routing occurs at several levels. For example, within a wide area network that has multiple physical connections between packet switches, the network itself is responsible for routing packets from the time they enter until they leave. Such internal routing is completely self-contained inside the wide area network. Machines on the outside cannot participate in decisions; they merely view the network as an entity that delivers packets.

Remember that the goal of IP is to provide a virtual network that encompasses multiple physical networks and offers a connectionless datagram delivery service. Thus, we will focus on Internet routing or IP routing. Analogous to routing within a physical network, IP routing chooses a path over which a datagram should be sent. The IP routing algorithm must choose how to send a datagram across multiple physical networks.

Routing in an Internet can be difficult, especially among computers that have multiple physical network connections. Ideally, the routing software would examine such things as network load, datagram length, or the type of service specified in the datagram header, when selecting the best path. Most Internet routing software is much less sophisticated, however, and selects routes based on fixed assumptions about shortest paths.

To understand IP routing completely, we must go back and look at the architecture of a TCP/IP Internet. First, recall that an Internet is composed of multiple physical networks interconnected by computers called routers. Each router has direct connections to two or more networks. By contrast, a host computer usually connects directly to one physical network. We know that it is possible, however, to have a multi-homed host connected directly to multiple networks.

Both hosts and routers participate in routing an IP datagram to its destination. When an application program on a host attempts to communicate, the TCP/IP protocols eventually generate one or more IP datagrams. The host must make a routing decision when it chooses where to send the datagrams. As Figure 4.6 shows, hosts must make routing decisions even if they have only one network connection.

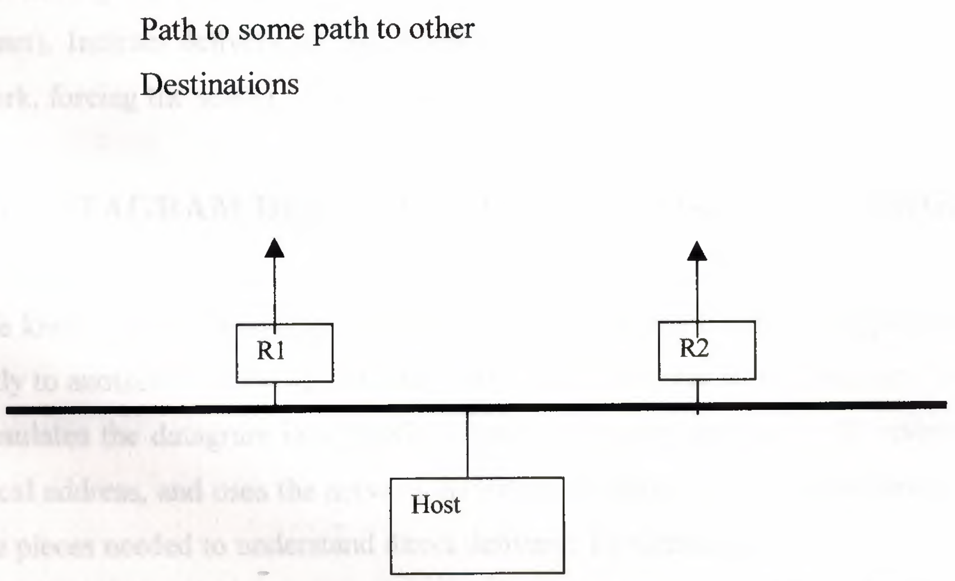


Figure 4.6 an example of a singly homed host that must route Datagrams. The host must choose to send a datagram either to router R1, or to router R2, because each router provides the best path to some destinations.

Of course, routers also make IP routing decisions (that is their primary purpose and the motivation for calling them routers). What about multi homed hosts? Any computer with multiple network connections can act as a router, and as we will see, multi homed hosts running TCP/IP have all the software needed for routing. Furthermore, sites that cannot afford separate routers sometimes use general-purpose timesharing machines as both hosts and routers (the practice is usually limited to university sites). And those of a router and sites that try to mix host and router functions on a single machine sometimes find that their multi-homed hosts engage in unexpected interactions. For now, we will distinguish hosts from routers and assume that hosts do not perform the router's function of transferring packets from one network to another.

4.8.3 DIRECT AND INDIRECT DELIVERY

Loosely speaking, we can divide routing into two forms: direct delivery and in direct delivery. Direct delivery, the transmission of a datagram from one machine across a single physical network directly to another, is the basis on which all Internet communication rests. Two machines can engage in direct delivery only if they both attach directly to the same underlying physical transmission system (e.g., a single Ethernet). Indirect delivery occurs when the destination is not on a directly attached network, forcing the sender to pass the datagram to a router for delivery.

4.8.4 DATAGRAM DELIVERY OVER A SINGLE NETWORK

We know that one machine on a given physical network can send a physical frame directly to another machine on the same network. To transfer an IP datagram, the sender encapsulates the datagram in a physical frame, maps the destination IP address into a physical address, and uses the network hardware to deliver it. Thus, we have reviewed all the pieces needed to understand direct delivery. To summarize:
Transmission of an IP datagram between two machines on a single physical network does not involve routers. The sender encapsulates the datagram in a physical frame, binds the destination IP address to a physical hardware address, and sends the resulting frame directly to the destination.

How does the sender know whether the destination lies on a directly connected network? The test is straightforward. We know that IP addresses are divided into a network-specific prefix and a host-specific suffix. To see if a destination lies on one of the directly connected networks, the sender extracts the network portion of the destination IP address and compares it to the network portion of its own IP addressees). A match means the datagram can be sent directly. Here we see one of the advantages of the Internet address scheme, namely:

Because the Internet addresses of all machines on a single network in-
Include a common network prefix, and because extracting that prefix
Can be done in a few machine instructions, testing whether a machine
Can be reached directly is extremely efficient.

From an Internet perspective, it is easiest to think of direct delivery as the final Step in any datagram transmission, even if the datagram traverses many networks and Intermediate routers. The final router along the path between the datagram source and its destination will connect directly to the same physical network as the destination. Thus, the final router will deliver the datagram using direct delivery. We can think of direct delivery between the source and destination as a special case of general purpose routing - in a direct route the datagram does not happen to pass through any intervening routers.

4.8.5 INDIRECT DELIVERY

Indirect delivery is more difficult than direct delivery because the sender must identify a router to which the datagram can be sent. The router must then forward the datagram on toward its destination network.

To visualize how indirect routing works, imagine a large Internet with many networks interconnected by routers but with only two hosts at the far ends. When one host wants to send to the other, it encapsulates the datagram and sends it to the nearest router. We know that it can reach a router because all physical networks are interconnected, so there must be a router attached to each one. Thus, the originating host can reach a router using a single physical network. Once the frame reaches the router, software extracts the encapsulated datagram, and the IP software selects the next router along the path towards the destination. The datagram is again placed in a frame and

Sent over the next physical network to a second router, and so on, until it can be delivered directly. These ideas can be summarized:

Routers in a TCPIIP Internet form a cooperative, interconnected Structure. Datagrams pass from router to router until they reach a Router that can deliver the datagram directly.

How can a router know where to send each datagram? How can a host know which router to use for a given destination? The two questions are related because they both involve IP routing. We will answer them in two stages, considering the basic table-driven routing algorithm in this chapter and postponing a discussion of how routers learn new routes until later.

CHAPTER 5

TCP/IP OVER ATM NETWORKS

5.1 INTRODUCTION

This chapter explores how TCP/IP design for connectionless networks, can be used over connection-oriented technology. We will see that TCP/IP is extremely flexible although a few of the address binding details change, most protocols remain unchanged. To make the discussion concrete and relate it to available hardware, we will use Asynchronous Transfer Mode (ATM) in all examples. ATM offers high speed can be used for both local area and wide area networks, and supports a variety of applications including real-time audio and video as well as a conventional data communication. This chapter expands the brief description in chapters, and covers additional details. In particular, the next sections describe the physical topology of an ATM network, the logical connectivity provided, ATM's connection paradigm, and the ATM protocol for data transfer.

Later sections explain the relationship between ATM and TCP/IP. They show an ATM host address relates to the host's IP address. They describe a modified form of the address resolution protocol (ARP) used to resolve an IP address to an ATM connection, and a modified form of inverse ARP used to help manage address bindings in a server. Most important, we will see how IP datagrams travel across an ATM network without IP fragmentation.

5.2 ATM HARDWARE

The basic component of an ATM network is a special-purpose electronic switch design to transfer data at extremely high speed. A typical small switch can connect between 16 and 32 computers. To permit data communication at high speeds, each connection between a computer and an ATM switch uses a pair of optical fibers.

Figure 5.1 illustrates the connection between a computer and an ATM switch.

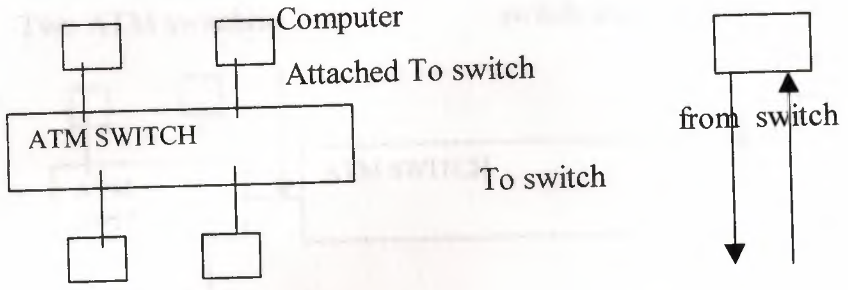


Figure 5.1 diagram of single ATM switch with four computers attached, and the details of a single Connection. A pair of optical fibers carries data to and from the switch

Physically, a host interfaces board plugs into a computer's bus. The interface hardware includes a light emitting diode (LED) or a
Miniature laser along with the circuitry needed to convert data into pulses of light that travel down the fiber to the switch. The interface also contains the hardware needed to sense pulses of light coming from the switch and converts them back into data bits in electronic form. Because a given fiber can carry light in only one direction,
a connection requires a pair of fibers to allow the computer to both send and receive data.

5.3 LARGE ATM NETWORKS

Although a single ATM switch has finite capacity, multiple switches can be interconnected to form a larger network. In particular, to connect computers at two sites to the same network, a switch can be installed at each site, and the two switches can then be connected. The connection between two switches differs slightly from the connection between a host computer and a switch. For example, interswitch connections can operate at higher speeds, and can use slightly modified protocols. Figure 5.2 illustrates the topology, and shows the difference between a network to network interface (NNI) and a user to network interfaces (UNI).

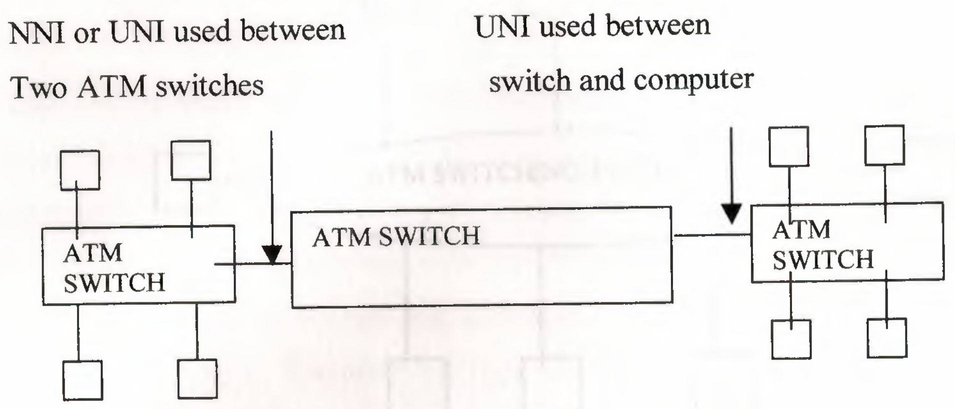


Figure 5.2 three ATM switches combined to form a large network. Although an NNI interface is designed for use between Switches, UNI connections can be used between ATM switches in a private network.

The destination between UNI and NNI arises because telephone companies designed ATM technology using the same paradigm as they use for the voice network. In general, a phone company that offers ATM data services to customers will also interconnect with other phone companies. The designers envisioned UNI as the interface between equipment at a customer's site and the switching equipment owned by the common carrier, and NNI as the interface between switches owned and operated by two different phone companies.

5.4 THE LOGICAL VIEW OF AN ATM NETWORK

To a computer attached to an ATM network, an entire fabric of ATM switches appears to be a homogenous network. Like the voice telephone system or a bridged Ethernet, ATM hides the details of physical hardware and gives the appearance of a single, physical network with many computers attached. For example, figure 5.3 illustrates how the ATM switching system in figure 5.3 appears logically to the eight Computers that are attached to it.

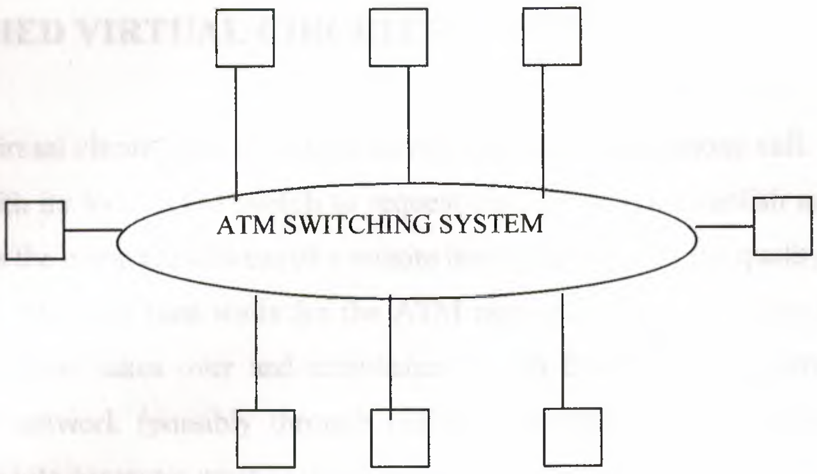


FIGURE 5.3 the logical view of the ATM switches in figure 5.3. ATM gives the appearance of a uniform network; any computer can communicate with any other computer.

Thus ATM provides the same general abstraction across homogenous ATM hardware that TCP/IP provides for heterogeneous system:

Despite a physical architecture that permits a switching fabric to contain Multiple switches, ATM hardware provides attached computers with the Appearance of a single, physical network. Any computer on the ATM Network can communicate directly with any other; the computers remain Unaware of the physical network structure.

5.5 THE TWO ATM CONNECTION PARADIGMS

ATM provides a connection-oriented interface to attached hosts. To reach a remote destination over an ATM network, a host must establish a connection, an abstraction that resembles a telephone call. ATM offers two forms of connections. The first is known as a Switched Virtual Circuit (SVC), and the second is known as a Permanent Virtual Circuit (PVC).

5.5.1 SWITCHED VIRTUAL CIRCUITS

A switched virtual circuit operates like a conventional voice telephone call. A host communicates with its local ATM switch to request that the switch establish an SVC. The host specifies the complete address of a remote host computer and the quality of Service required. The host then waits for the ATM network to create a circuit. The ATM signaling system takes over and establishes a path from the originating host, across the ATM network (possibly through multiple switches), to the remote host computer. The remote computer must agree to accept the virtual circuit.

During signaling, each ATM switch along the path examines the quality of service requested for the circuit. If it agrees to forward data, a switch records information about the circuit and sends the request to the next switch along the path. Such an agreement requires a commitment of hardware and software resources at each switch.

When signaling completes, the local ATM switch reports success to both ends of the switched virtual circuit.

The ATM UNI interface uses a 24-bit integer to identify each virtual circuit. When a host creates or accepts a new virtual circuit, the local ATM switch assigns an identifier to the circuit. A packet transmitted across an ATM network contains neither a source nor destination address. Instead, a host labels each outgoing packet and the Switch labels each incoming packet with a circuit identifier.

Note that we have skipped over several details of signaling, including the protocol a host uses to request a new circuit and the protocol a switch uses to inform the host that a connection request has arrived from a remote host. Furthermore, we have omitted a few details that are important in practice. For example, two-way communication requires resources to be reserved along the reverse path as well as the forward path.

5.6 PATHS, CIRCUITS, AND IDENTIFIERS

ATM assigns a unique integer identifier to each circuit a host has open; the host uses the identifier when performing 1/0 operations or when closing the circuit. A circuit identifier is analogous to a descriptor that a program uses to perform 1/0. Like an 1/0 descriptor, a circuit identifier is short compared to the information needed to create a circuit. Also like an 1/0 descriptor, a circuit identifier only remains valid while the circuit is open. Furthermore, a circuit identifier is meaningful only across a single hop - the circuit identifiers obtained by hosts at the two ends of a given virtual circuit usually differ. For example, the sender may be using identifier 17 while the receiver uses identifier 49; each ATM switch translates the circuit identifier in a packet as the packet flows from one host to the other.

Technically, a circuit identifier used with the UNI interface consists of a 24-bit integer divided into two fields. Figure 5.4 shows how ATM partitions the 24 bits into an 8-bit virtual Path identifier- (VPI) and a 16-bit virtual circuit identifier (VCI). Often, the entire identifier is referred to as a VPIIVCI pair.

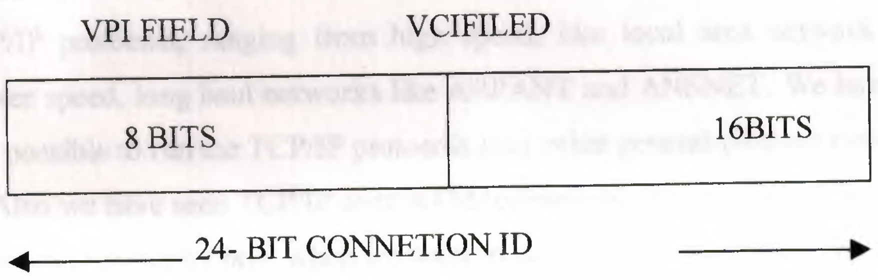


Figure 18.4 the 24-bit connection identifier used with UNI. The Identifier is divided into virtual path and virtual circuit parts.

The motivation for dividing a connection identifier into VPI and VCI fields is similar to the reasons for dividing an IP address into network fields.

Conclusion

In my project I have reviewed Internetworking with TCP/IP protocol, the TCP/IP protocols are extremely flexible in that almost any underlying technology can be used to transfer TCP/IP traffic.

TCP/IP protocols are as language between the computers in the world, the fundamental service provided by TCP/IP Internet software is connectionless unreliable, best-effort packet delivery. The Internet protocol (IP) formally specifies the format of Internet packets, also we can have several network hardware technologies used by the TCP/IP protocols, ranging from high speed, like local area network as Ethernet to slower speed, long haul networks like ARPANT and ANSNET. We have also seen that it is possible to run the TCP/IP protocols over other general-purpose network protocols.

Also we have seen TCP/IP over ATM technology, because ATM is a connection-Oriented technology now when we want to do connection between two computers must establish virtual circuit through the network before they can transfer data; that means to Install TCP/IP protocol to the network.

TCP/IP uses 32-bit binary addresses as universal machine identifiers. This called internet or IP address, because the IP address encodes network identification as well as the identification of a specific host on that network also an important property of IP address is that they refer to network connection. The Internet addressing scheme is that the form includes an address for a specific host.

Now, there are also standards protocols that specify how data is represented when being transferred from one machine to another. Protocols specify how the transfer occurs, the idea of protocols layering is fundamental because it provides a conceptual framework for protocol design.

Conclusion

In my project I have reviewed Internetworking with TCP/IP protocol, the TCP/IP protocols are extremely flexible in that almost any underlying technology can be used to transfer TCP/IP traffic.

TCP/IP protocols are as language between the computers in the world, the fundamental service provided by TCP/IP Internet software is connectionless unreliable, best-effort packet delivery. The Internet protocol (IP) formally specifies the format of Internet packets, also we can have several network hardware technologies used by the TCP/IP protocols, ranging from high speed, like local area network as Ethernet to slower speed, long haul networks like ARPANT and ANSNET. We have also seen that it is possible to run the TCP/IP protocols over other general-purpose network protocols.

Also we have seen TCP/IP over ATM technology, because ATM is a connection-Oriented technology now when we want to do connection between two computers must establish virtual circuit through the network before they can transfer data; that means to Install TCP/IP protocol to the network.

TCP/IP uses 32-bit binary addresses as universal machine identifiers. This called internet or IP address, because the IP address encodes network identification as well as the identification of a specific host on that network also an important property of IP address is that they refer to network connection. The Internet addressing scheme is that the form includes an address for a specific host.

Now, there are also standards protocols that specify how data is represented when being transferred from one machine to another. Protocols specify how the transfer occurs, the idea of protocols layering is fundamental because it provides a conceptual framework for protocol design.

REFERENCES

[1] Douglas E, David l and Stevens ., " design, implementation and internals", internetworking with TCP/IP protocol, vol.II, 1995

[2] Douglas comer., "principles, protocols, Architecture", internetworking with TCP/IP protocol, vol.II, 1993

[3] Carcia-Aceves,.J.J.,Stahl, and c. A. Ward,eds. Internet protocol handbook :the domain name server system(DNS) handbook.menlo park,CA:SRI international, network information systems center;1989