Computer Networks and Networking

Version 2.01
May 2005

Contents

Chapter 1

Course Outline

1.1 How to use this document

1.1.1 Do

- Use this document when seeking an alternative explanation to concepts covered in lectures.
- Summarize relevant material from your lecture notes, and this document to use for studying for tests and exams.
- Use your judgement to reconcile differences between the contents of this document and the material presented in lectures. This material comes from a range of sources, and some systems have evolved over time.
- Try to answer the questions, and solve the problems associated with the material in this document. They are intended to encourage you to think about the issues involved in this area of computer science.

1.1.2 Don't

- Expect these notes to contain all information, explanations, or examples given during lectures.
- Expect to find a concise summary of only relevant points in these notes. This documents contains extra material which may not be used in the course that you are given.
- Blindly repeat what is written in this document. Be prepared to select material based on the context, and present it in a manner that demonstrates understanding of the concepts.
- Read the answers to the exercises without at least first thinking about them. Not being able to identify whether a problem is relevant to the course you are taking is also a bad sign.

1.2 Outcomes

The overall outcomes of this course are:

1. Competence in applying knowledge to solve problems and answer questions relating to computer networking.
2. The ability to communicate with others working in computer networking in a professional manner.
3. Achievement of an understanding of the technology and issues relating to computer networking, and an appreciation of the need for life-long learning.

1.2.1 Specific Outcomes

To achieve competence in applying knowledge to solve problems and answer questions relating to computer networking, the specific outcomes related to this course are:

1. Demonstrate knowledge and understanding of the information, concepts and principles applicable to computer networking.

2. Be able to construct, analyze and critically comment on network protocols and protocol design issues.

3. Collect, present and analyze data relating to the performance of computer networks.

4. Work effectively with others situated on remote workstations, using a computer network.

5. Communicate an understanding of computer networking and the results of laboratory work.

6. Use computer networking technology with due appreciation of the social (particularly security) issues involved.

To achieve competence in communicating with others working in computer networking in a professional manner, the specific outcomes are:

1. Ability to communicate ideas relating to computer networks, specifically protocol designs, network layout and network models in appropriate written and diagrammatic form.

To achieve of an understanding of the technology and issues relating to computer networking, and an appreciation of the need for life-long learning, the specific outcomes are:

1. Be able to classify the networking technologies presented during the course according to a consistent and extensible framework.

2. Demonstrate the ability to access new information from appropriate sources, particularly the Internet, and reference such material correctly.

3. Incorporate such new information into solutions to problems involving computer networks.

4. Appreciate that there is a need for life-long learning to remain current in the field of computer networking.

1.2.2 Assessment criteria

The following should be demonstrated during exercises undertaken during the course:

1. A detailed understanding, both in theory and practice, of the principles, theories and facts related to computer networking.

2. Basic research, creative problem solving and critical thinking skills, including use of available resources to gain access to information.

3. Well developed communicative skills and the ability to convey information in an appropriate manner for communicating with other computer networking professionals.

Assessment takes place through theoretical and practical exercises during the course, in the form of a test during the course and through an examination at the end of the course.

1.3 Objectives

This course:

1. Provides information on the concepts and principles applicable to computer networking.
2. Describes a framework for structuring knowledge related to computer networking.
3. Encourages comparison and evaluation of the techniques presented.
4. Covers strategies for modelling, measuring and analyzing computer networks protocols and network design.
5. Provides case studies and examples to illustrate the application of the material, and to demonstrate how it can be communicated.
6. Provides a range of exercises intended to encourage both practical experience with the material, and further investigation into the implications of the content.

1.4 Overview of the course material

Computer networking involves the use of a wide range of hardware and software technologies. The diversity of the technologies used is continuously increasing, making it essential that a suitable framework for classifying them be built in order to appreciate the specialized role of each.

Chapter 2 introduces some of the criteria that can be and are used to categorize computer networking technologies. The relative merits of different networking strategies are discussed. It also defines a core set of networking terminology which is used throughout the remainder of the course.

This classification scheme is continued in Chapter 3. This chapter introduces the standard used by computer networking professionals for classification of computer networking technology by function (in terms of the OSI model), rather than by form. This functionality is related to view of a network as a stack of network protocols, each which performs a specific task. Network protocol design issues are identified during this process, and current strategies used by existing protocols are described and compared.

Having accepted the layered approach to network protocol design, the lower three layers of the OSI model are explored in detail in subsequent chapters.

Chapter 4 describes techniques for dealing with transmission over computer networks at the (lowest) physical layer. This layer deals with electrical signalling issues: methods for encoding bits, bytes and frames of information. It also introduces some of the error checking and correcting technologies which are used at the Data Link layer.

A range of Data Link technologies are presented in Chapter 5. This chapter represents a substantial portion of the course, providing both details about each specific Data Link technology, as well as an opportunity to compare and classify the diverse strategies applied in this layer.

Protocols found at the network layer (and above) are introduced in Chapter 6. The various strategies employed by the Internet protocols are covered in the most detail. An overview is given of the most likely technologies that a computer networking professional is likely to encounter.

Chapter 7 is provided as a resource for those interested in the details behind the operation of networking routers and bridges.

Having provided the knowledge of the information, concepts and principles applicable to computer networking, subsequent chapters investigate application of this knowledge to improve understanding, show applications and provide techniques for designing and evaluating practical networks.

The principles of network design are covered in Chapter 8. Evaluation criteria for network designs are presented and the relative benefits of different strategies are discussed. Case studies of network designs under different scenarios are presented. The emphasis in these case studies is on clearly presenting the network layout, particularly in diagrammatic form, and in critical analysis of the merits of the design.

Established techniques for network modelling and analysis are described in Chapter 9. A tool (Petri Nets) for extracting the key features of a network or protocol design is presented. Analysis strategies that

can measure properties of the model are demonstrated, and the implications of the results in the context of the original system are discussed. Simulation is presented as a technique for obtaining further results from system models. Issues relating to the accuracy of the simulation and its results are covered.

Issues relating to the use of computer networking in practice are the topics of the remaining chapters. Network management is covered in Chapter 10. This topic concentrates on the most popular standard for network management and covers issues of network management architectures from the point of view of network design, and of protocol design. Relative merits of this approach are discussed. A comparison with other management strategies shows key factors that influence the adoption of networking technologies.

Network security is of great importance to the professional working in this field, and relevant topics are introduced in Chapter 11. Security issues affecting hosts running specific operating systems and applications are discussed. Techniques for extending network design to include security considerations are covered. Finally some material is provided on cryptography, particularly where it relates to technologies used on the Internet

Chapter 2

Network Principles

2.1 Chapter Structure

2.1.1 Outcomes

1. Demonstrate knowledge and understanding of computer network constructs and configurations.
2. Be able to analyze network configurations and critically comment on relative merits of specific approaches.
3. Ability to use appropriate terminology for discussing computer networks.
4. Be able to classify the networking technologies presented during the course according to a consistent and extensible framework.
5. Identify recent developments in the field of computer networking.

2.1.2 Objectives

This chapter:

1. Describes network constructs such as: client-server and peer-to-peer.
2. Describes network classification criteria, such as: size, topology, wiring type, multiplexing technique.
3. Evaluates various configurations under each criterion and discusses relative merits of each.
4. Introduces terms commonly used in computer networking, including: LAN, WAN, Bus, Ring, Star, Mesh, TDMA, FDMA, CDMA, circuit switching, packet switching, virtual circuits, datagram.
5. Describes both past and present technologies used as solutions to networking problems.

2.2 Computer Networks

A computer network can be defined as a network of data processing nodes that are interconnected for the purpose of data communication, or alternatively as a communications network in which the end instruments are computers.

The nodes that one may find on a network can include:

- Servers: computers used to store the shared information and have all the other computers reference that information over a network.

- Clients: computers on a network that use, but do not provide, network resources.
- Peers: computers on a network that that both use and provide network resources.

Networks are often broadly classified in terms of the typical communication patterns that one may find on them. Three common types of networks are:

1. Server-based (client/server) - contain clients and the servers that support them
2. Peer (peer-to-peer) - contain only clients, no servers, and use network to share resources among individual peers.
3. Hybrid - client/server that also contains peers sharing resources (most common for corporations).

Client/Server networks offer a single strong central security point, with central file storage which provides multi-user capability and easy backup. It also gives the ability to pool the available hardware and software, lowering overall costs. Optimized dedicated servers can make networks run faster. Dedicated server hardware is usually expensive, and the server must run an often expensive network operating system software. A dedicated network administrator is usually required.

Servers may be classified as:

1. File Servers - offer services that allow network users to share files and provide central file management services (such as backups).
2. Print Servers - manage and control printing on a network, allowing users to share printers.
3. Application Servers - allow client machines to access and use extra computing power and expensive software applications that reside on the server.
4. Message Servers - data can pass between users on a network in the form of graphics, digital video or audio, as well as text and binary data (for example: e-mail).
5. Database Servers -provide a network with powerful database capabilities that are available for use on relatively weaker client machines.

Typical communication in a client/server system involves the client sending a request for data, the server waiting for requests, processing received requests and sending responses, and the clients waiting for, and using, the response.

Peer networks are defined by a lack of central control over a network. Users share resources, disk space, and equipment. The users to control resource sharing, and so there may be lower security levels, and no trained administrator. Since there is no reliance on other computers (server) for their operation such networks are often more tolerant to single points of failure. Peer networks place additional load on individual PCs because of resource sharing. The lack of central organization may make data hard to find, backup or archive.

Hybrid networks can combine the advantages and disadvantages of both of the above types.

These network architectures can be compared with the pre-network host-based model. During the old days of yore, when computers first came out, they were huge clunky things servicing people sitting at dumb terminals. In a host based system, the dumb terminals are just that, dumb. They didn't think. They listen and they do, something like a robot really, without thinking. The host (central mainframe computer) does all the thinking for them. Networks could be employed to interconnect two or more mainframe computers. Terminals could connect only to the mainframe, and never to each other. In a client-server environment, the clients can do some processing on their own as well, without taxing the server. In a peer to peer environment, clients can be connected to one another.

2.3 Classification of Networks

A computer network is created when data communication channels link several computers and other devices, such as printers and secondary storage devices. Computer networks can be classified according to a number of criteria (see Table 2.1).

Network Classification			
Size	Topology	Cable	Signal Transmission
LAN	Bus	Coaxial	TDM
MAN	Star	UTP	STDM
WAN	Ring	STP	Circuit Switching
	Star-Wired-Ring	Fiber	Packet Switching
	Tree		- Datagram
	Mesh		- Virtual Circuit

Table 2.1: Network classification criteria.

2.3.1 Network Size

2.3.1.1 Local Area Networks

A Local Area Network (LAN) is a communications network that serves users within a confined geographical area. Specifically it has the properties:

- a limited-distance (typically under a few kilometers)
- high-speed network (typically 4 to 100 Mbps)
- supports many computers (typically two to thousands).
- a very low error rate
- owned by a single organization
- most commonly uses ring or bus topologies

The message transfer is managed by a transport protocol such as TCP/IP and IPX. The physical transmission of data is performed by the access method (Ethernet, Token Ring, etc.) which is implemented in the network adapters that are plugged into the machines. The actual communications path is the cable (twisted pair, coax, optical fiber) that interconnects each network adapter.

2.3.1.2 Wide Area Networks

A Wide Area Network (WAN) is a communications network that covers a wide geographic area, such as state or country. Contrast this to a LAN (local area network) which is contained within a building or complex, and a MAN (metropolitan area network) which generally covers a city or suburb. The WAN can span any distance and is usually provided by a public carrier. You get access to the two ends of a circuit; the carrier does everything in between–which is typically drawn as a "grey cloud," since you don't know (or usually care) how the carrier implements it.

2.3.2 Topology

A network configuration is also called a network topology. A network topology is the shape or physical connectivity of the network. The network designer has three major goals when establishing the topology of a network:

1. Provide the maximum possible reliability: provide alternative routes if a node fails and be able to pinpoint the fault readily, deliver user data correctly (without errors) and recover from errors or lost data in the network.

2. Route network traffic through the least cost path within the network: minimizing the actual length of the channel between the components and providing the least expensive channel option for a particular application.

3. Give the end users the best possible response time and throughput.

The topology of the network can be viewed in two ways:

1. The topology as seen from the layout of the cable, or the route followed by the electrical signals. This is the **physical** topology.

2. The connections between nodes as seen by data traveling from one node to another - reflects the network's function, use, or implementation without regard to the physical interconnection of network elements. This is the **logical** topology, and may be different from the physical topology. See section 2.3.2.4 or Figure 2.1 for an example.

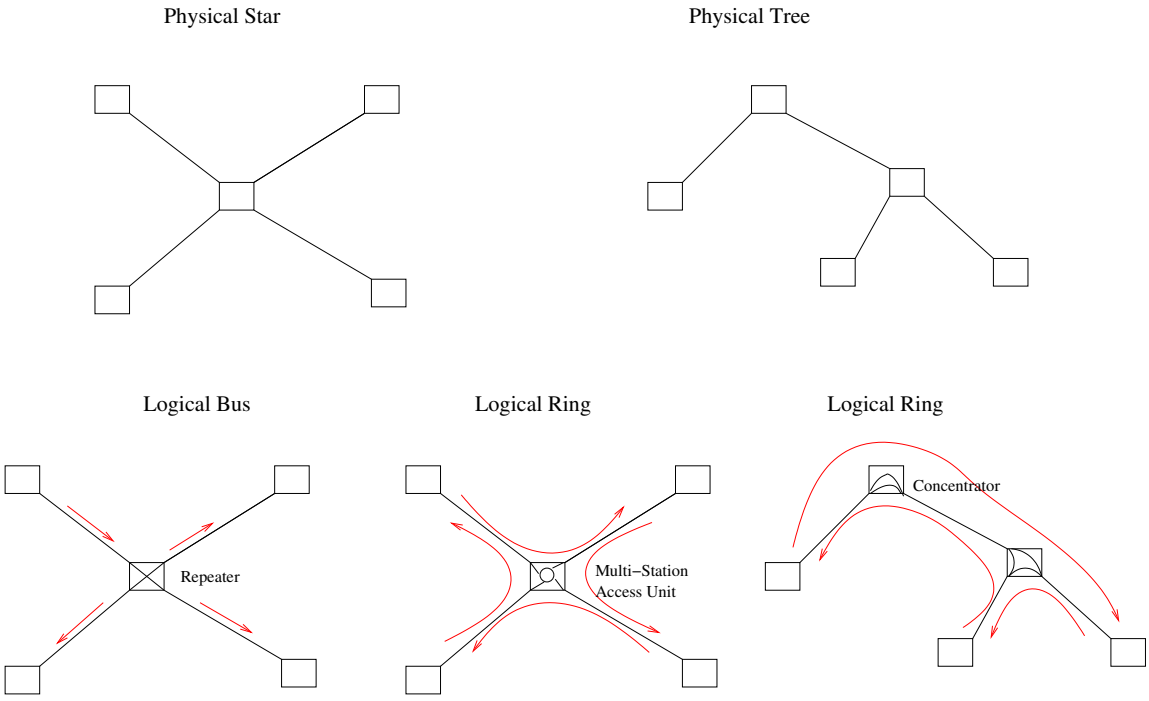


Figure 2.1: Physical versus Logical Topologies

A range of different topologies are common, with the properties as summarized in Table 2.2.

2.3.2.1 Bus

In a bus topology each node (computer, server, peripheral etc.) attaches directly to a common cable. This topology most often serves as the backbone for a network. In some instances, such as in classrooms or labs, a bus will connect small workgroups. Since a hub is not required in a bus topology, the set-up cost is relatively low. However, this topology's wiring scheme is unstructured (without a central point of concentration) making it difficult to troubleshoot. Often if one PC goes down, the whole network can shut down.

Usually the bus must be *terminated*. Termination is the process of stopping signals sent through a network. Without termination, signals bounce back and forth, causing a log jam over a network.

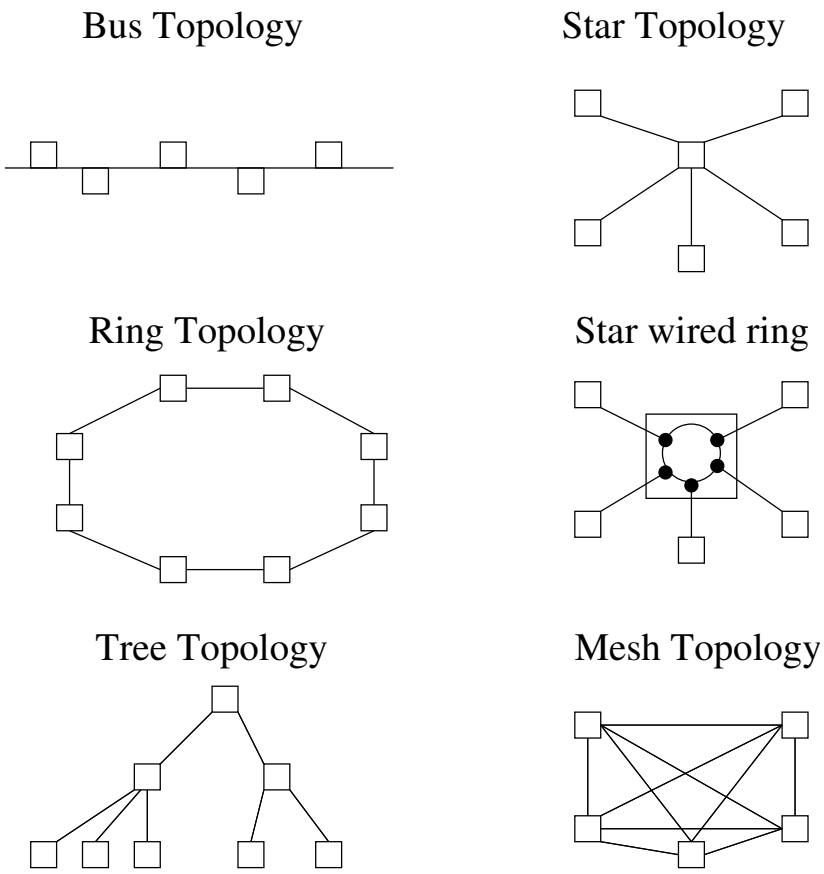


Figure 2.2: Various network topologies

Bus networks are simple, easy to use, and reliable. They require the least amount of cable and are easy to extend. Repeaters can be used to boost signal and extend bus.

Heavy network traffic can slow a bus considerably. Each connection weakens the signal, causing distortion among too many connections.

2.3.2.2 Star

A star topology, on the other hand, is relatively easy to troubleshoot due to its structured wiring scheme. With this topology, each node has a dedicated set of wires connecting it to a central network hub. The failure of one connection will not usually affect the others. And, since all traffic passes through the hub, the hub becomes a central point for isolating network problems and gathering network statistics.

The star topology can have a number of different transmission mechanisms, depending on the nature of the central hub.

- Broadcast Star Network: The hub receives and resends the signal to all of the nodes on a network.
- Switched Star Network: The hub sends the message to only the destination node.
- Active Hub (Multi-port Repeater): Regenerates the electric signal and sends it to all the nodes connected to the hub.
- Passive Hub: Does not regenerate the signal; simply passes it along to all the nodes connected to the hub.

	Reliability	Cost	Response
Bus	Cable break can segment network, or prevent transmission	Single cable, low cost	Shared medium, limits performance
Star	Easy to troubleshoot, loss of one does not affect others	Cost of wiring and central hub if required	Sharing, or switching possible
Ring	One failure destroys network	Single cable, with repeaters at each station	Single medium limits performance
Star-wired Ring	Breaks can be isolated	Longer wires, and hub	As for ring
Tree	Root nodes can be a vulnerability	May need routing hardware	Root node can be a bottleneck
Mesh	Quite immune to individual cable breaks	Wiring expensive	Alternative routes available

Table 2.2: Summary of the properties of different topologies.

- Hybrid Star Network: Placing another star hub where a client node might otherwise go.

Star networks are easy to modify and one can add new nodes without disturbing the rest of the network. Intelligent hubs provide for central monitoring and managing. Often there are facilities to use several different cable types with hubs.

Central hub failure will lead to total network failure. They are also costly to cable since all network cables must be pulled to one central point.

2.3.2.3 Ring

A ring topology features a logically closed loop of cable - a ring. Data packets travel in a single direction around the ring from one network device to the next. Each network device acts as a repeater, meaning it regenerates the signal. If one device fails, the entire network goes down. This disadvantage gave rise to a hybrid topology referred to as the star-wired ring.

2.3.2.4 Star-wired Ring

The star-wired ring has essentially replaced the ring topology in practical use. Networks based on star-wired ring topologies have nodes radiating from a wiring center or hub. The hub acts as a logical ring with data packets traveling in sequence from port to port. Just like a star topology, if one node fails, the network will continue to operate.

2.3.2.5 Tree (Hierarchy)

The hierarchical topology is one of the more common topologies found today. The software to control the network is relatively simple and the topology provides a concentration point for control and error resolution. The node at the highest point in the hierarchy usually controls the network.

Whilst this type of network is attractive for its simplicity it does present a potential significant bottleneck problem. In some instances the uppermost node will control all the traffic. Not only can this cause a bottleneck, but it can also present reliability problems if this node fails.

2.3.2.6 Mesh

The mesh topology has been used more frequently in recent years. Its primary attraction is its relative immunity to bottlenecks and channel/node failures. Due to the multiplicity of paths between nodes, traffic can easily be routed around failed or busy nodes. Given that this approach is very expensive in comparison to

other topologies, some users will still prefer the reliability of the mesh network to that of others (especially for networks that only have a few nodes that need to be connected together).

2.3.3 Cable type

Cable is what physically connects network devices together, serving as the conduit for information traveling from one computing device to another. The type of cable you choose for your network will be dictated in part by the network's topology, size and media access method. Small networks may employ only a single cable type, whereas large networks tend to use a combination.

In Project 802, the IEEE established specifications for cables carrying Ethernet signals. 10Base5, 10Base2, 10Base-T and 10Base-F refer to thick coaxial, thin coaxial, unshielded twisted-pair and fiber-optic cables respectively.

The "10" refers to the Ethernet transmission speed - 10 Mbps. The "Base" refers to baseband (single communications channel on each cable). Originally, the last character referred to the maximum cable distance in hundreds of meters. This naming convention changed, however, with the introduction of 10Base-T and 10Base-F. In these instances, the T and F refer to the cable types (twisted-pair and fiber-optic).

2.3.3.1 Coaxial Cable

Coaxial cable includes a copper wire surrounded by insulation, a secondary conductor that acts as a ground, and a plastic outside covering (see Figure 2.3). Because of coaxial cable's two layers of shielding, it is relatively immune to electronic noise, such as motors, and can thus transmit data packets long distances. Coaxial cable is a good choice for running the lengths of buildings (in a bus topology) as a network backbone.

Local area networks (LANs) primarily use two sizes of coaxial cable, commonly referred to as thick and thin. Thick coaxial cable can extend longer distances than thin and was a popular backbone (bus) cable in the 1970s and 1980s. However, thick is more expensive than thin and difficult to install. Today, thin (which looks similar to a cable television connection) is used more frequently than thick.

2.3.3.2 Twisted-Pair Cable

Twisted-pair cable consists of two insulated wires that are twisted around each other and covered with a plastic casing. It is available in two varieties, unshielded and shielded. Unshielded twisted-pair (UTP) is similar in appearance (see Figure 2.4) to the wire used for telephone systems. UTP cabling wire is grouped into categories, numbered 1-5. The higher the category rating, the more tightly the wires are twisted, allowing faster data transmission without crosstalk. Since many buildings are pre-wired (or have been retrofitted) with extra UTP cables, and because UTP is inexpensive and easy to install, it has become a very popular network media over the last few years.

Shielded twisted-pair cable (STP) adds a layer of shielding to UTP. Although STP is less affected by noise interference than UTP and can transmit data further, it is more expensive and more difficult to install.

2.3.3.3 Fiber-Optic Cable

Fiber-optic cable is constructed of flexible glass and plastic. It transmits information via photons, or light. More resistant to electronic interference than the other media types, fiber-optic is ideal for environments with a considerable amount of noise (electrical interference). Furthermore, since fiber-optic cable can transmit signals further than coaxial and twisted-pair, more and more educational institutions are installing it as a backbone in large facilities and between buildings. The cost of installing and maintaining fiber-optic cable remains too high, however, for it to be a viable network media connection for classroom computers.

2.3.4 Signal Transmission Mechanisms

The way data are delivered through networks requires solutions to several problems:

- Methods for carrying multiple data streams on common media.

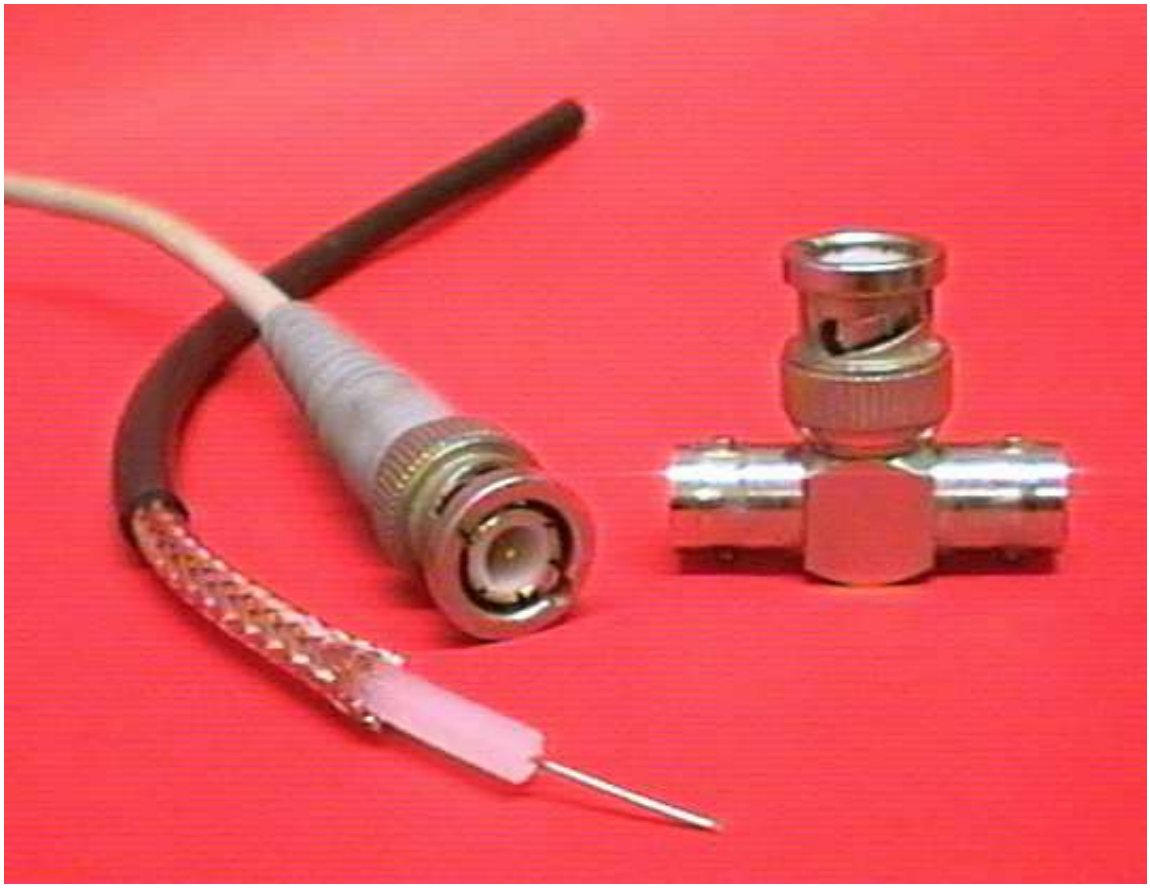


Figure 2.3: Coaxial cable, with BNC end connector, and T piece.

- Methods for switching data through paths on the network.
- Methods for determining the path to be used.

2.3.4.1 Multiplexing

LANs generally operate in baseband mode, which means that a given cable is carrying a single data signal at any one time. The various devices on the LAN must take turns using the medium. This generally is a workable approach for LANs, because LAN media offer high performance at low cost.

Long-distance data communication media are expensive to install and maintain, and it would be inefficient if each media path could support only a single data stream. WANs, therefore, tend to use broadband media, which can support two or more data streams. Increasingly, as LANs are expected to carry more and different kinds of data, broadband media are being considered for LAN as well.

To enable many data streams to share a high-bandwidth medium, a technique called multiplexing is employed. A wide range of different multiplexing strategies are possible.

2.3.4.1.1 Time Division Multiple Access (TDMA) The signals-carrying capacity of the medium is divided into time slots, with a time slot assigned to each signal, a technique called Time-Division Multiplexing (TDM), illustrated in Figure 2.5. Because the sending and receiving devices are synchronized to recognize the same time slots, the receiver can identify each data stream and re-create the original signals. The sending device, which places data into the time slots, is called a multiplexer or mux. The receiving

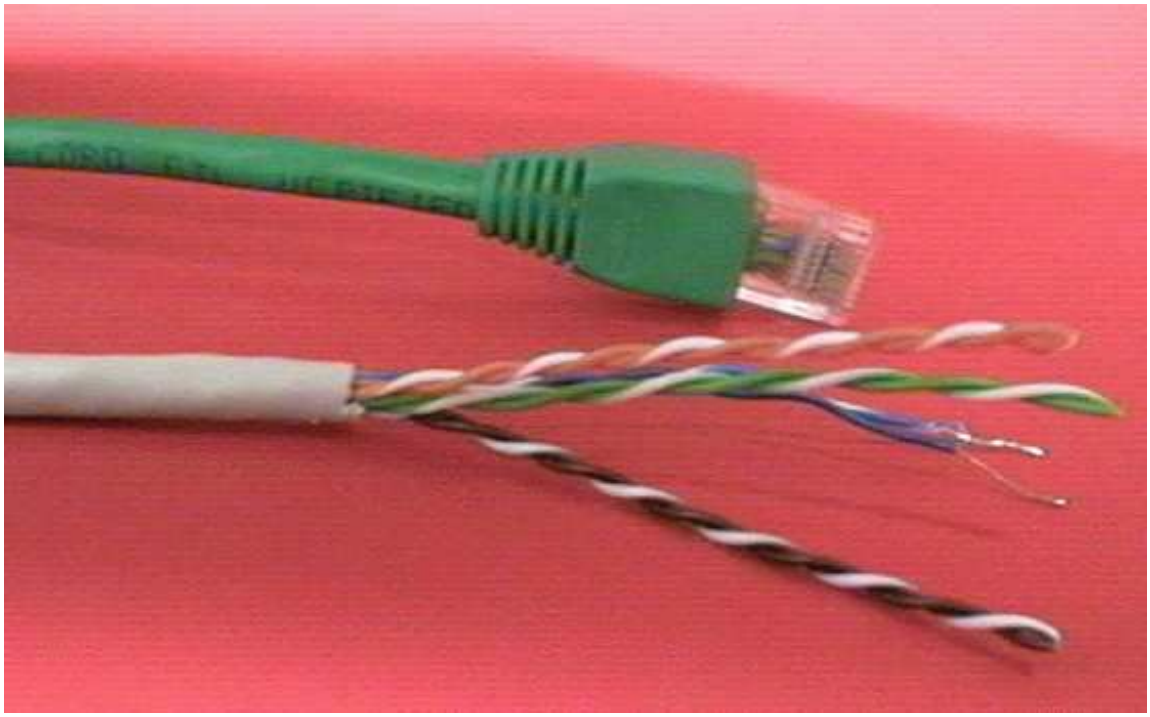


Figure 2.4: UTP cable.

device is called a demultiplexer or demux. TDM can be inefficient. If a data stream falls silent, its time slots are not used and the media bandwidth is underutilized.

A more advanced technique is statistical time-division multiplexing. Time slots are still used, but some data streams are allocated more time slots that others. An idle channel, D, is allocated no time slots at all. A device that performs statistical TDM often is called a stat-MUX.

2.3.4.1.2 Frequency Division Multiple Access (FDMA) Frequency division multiplexing requires that the different signals be assigned to separate frequency bands. There they can modulate the intensity (amplitude modulation) of the carrier signal, or create small variations in the frequency (frequency modulation) of the carrier signal. All the different carriers can be carried simultaneously over the cable, and separated on the far end to allow demultiplexing to occur.

Similar in principle to FDM is the idea of Wavelength Division Multiplexing. In this case each frequency used corresponds to a wavelength of light. This is of particular value when working with fiber optics, since each signal will be multiplexed over one of the colours of light being transmitted down the cable, by using it to control a laser of its particular colour. A number of mechanisms (such as diffraction gratings) can be used to separate and redirect the various colours, allowing the different signals to be routed without ever having to be converted back into electrical signals.

2.3.4.1.3 Code Division Multiple Access (CDMA) CDMA is a form of spread spectrum communications. Applications for commercial spread spectrum range from wireless LANs, to integrated bar code scanner/palmtop computer/radio modem devices for warehousing, to digital dispatch, to digital cellular telephone communications, to information society city/area/state or country wide networks for passing faxes, computer data, email, or multimedia data.

Spread-spectrum radio communications has long a favorite technology of the military because it resists jamming and is hard for an enemy to intercept. Spread-spectrum signals are distributed over a wide range of frequencies and then collected onto their original frequency at the receiver, and are so inconspicuous as

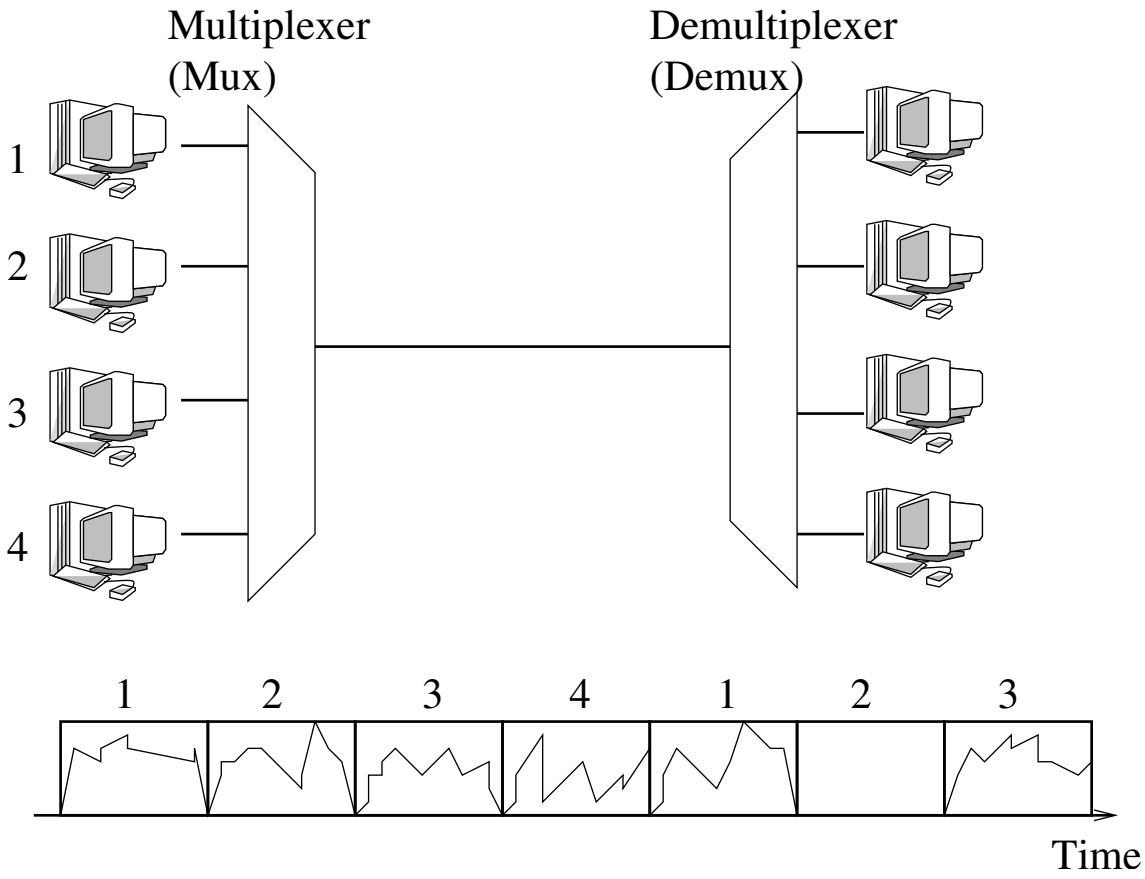


Figure 2.5: Time Division Multiplexing.

to be transparent. Just as they are unlikely to be intercepted by a military opponent, so are they unlikely to interfere with other signals intended for business and consumer users – even ones transmitted on the same frequencies. Such an advantage opens up crowded frequency spectra to vastly expanded use.

To qualify as a spread spectrum signal, two criteria should be met:

1. The transmitted signal bandwidth is much greater than the information bandwidth.
2. Some function other than the information being transmitted is employed to determine the resultant transmitted bandwidth.

The wide-band frequency spectrum desired is generated in a frequency hopping system. It does just what its name implies. That is, it hops from frequency to frequency over a wide band. The specific order in which frequencies are occupied is a function of a code sequence, and the rate of hopping from one frequency to another is a function of the information rate.

Frequency hopping is the easiest spread spectrum modulation to use. Any radio with a digitally controlled frequency synthesizer can, theoretically, be converted to a frequency hopping radio. This conversion requires the addition of a pseudo noise (PN) code generator to select the frequencies for transmission or reception. De-hopping in the receiver is done by a synchronized pseudo noise code generator that drives the receiver's local oscillator frequency synthesizer.

The use of these special pseudo noise codes in spread spectrum communications makes signals appear wide band and noise-like. It is this very characteristic that makes these signals possess the quality of Low Probability of Intercept. Signals are hard to detect on narrow band equipment because the signal's energy is spread over a bandwidth of maybe 100 times the information bandwidth.

The spread of energy over a wide band, or lower spectral power density, makes SS signals less likely to interfere with narrow-band communications. Narrow band communications, conversely, cause little to no interference to SS systems because the correlation receiver effectively integrates over a very wide bandwidth to recover an SS signal.

Besides being hard to intercept and jam, spread spectrum signals are hard to exploit or spoof. Signal exploitation is the ability of an enemy (or a non-network member) to listen in to a network and use information from the network without being a valid network member or participant. Spoofing is the act of falsely or maliciously introducing misleading or false traffic or messages to a network.

Spread spectrum technology is being widely used in modern telecommunications. Third generation cellular phones are based on this form of multiplexing.

2.3.4.2 Switching Data

On an internetwork, data units must be switched through the various intermediate devices until they are delivered to their destination. Two contrasting methods of switching data are commonly used: Circuit switching and packet switching. Both are used in some form by protocols in common use.

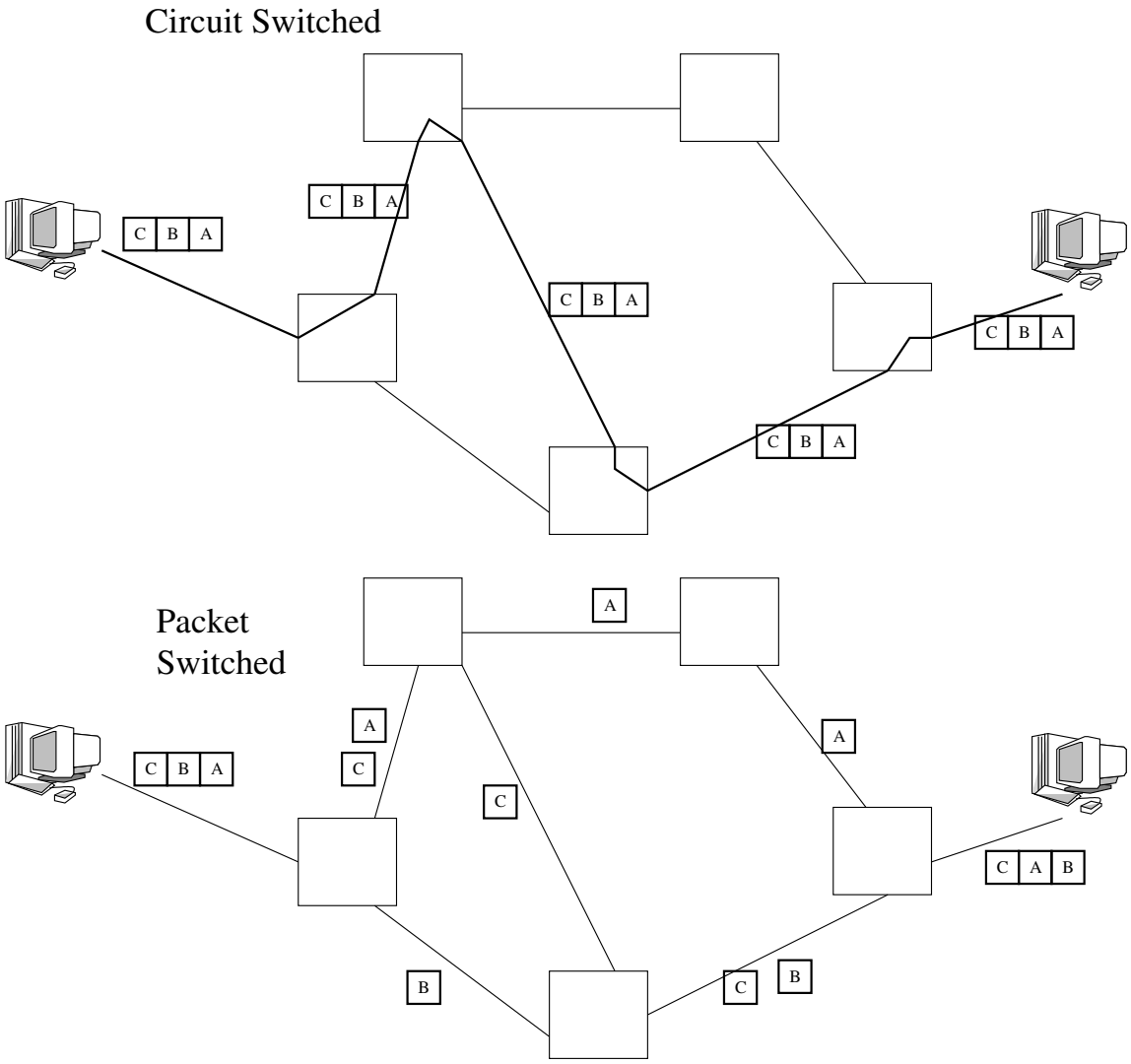


Figure 2.6: Circuit and Packet Switching.

2.3.4.2.1 Circuit Switching When two devices negotiate the start of a dialog, they establish a path, called a circuit, through the network, along with a dedicated bandwidth through the circuit (see Figure 2.6). After establishing the circuit, all data for the dialog flow through that circuit. The chief disadvantage of circuit switching is that when communication takes place at less than the assigned circuit capacity, bandwidth is wasted. Also, communicating devices can't take advantage of other, less busy paths through the network unless the circuit is reconfigured.

Circuit switching does not necessarily mean that a continuous, physical pathway exists for the sole use of the circuit. The message stream may be multiplexed with other message streams in a broadband circuit. In fact, sharing of media is the more likely case with modern telecommunications. The appearance to the end devices, however, is that the network has configured a circuit dedicated to their use.

End devices benefit greatly from circuit switching. Since the path is pre-established, data travel through the network with little processing in transit. And, because multi-part messages travel sequentially through the same path, message segments arrive in an order and little effort is required to reconstruct the original message.

2.3.4.2.2 Packet Switching Packet switching takes a different and generally more efficient approach to switching data through networks. Messages are broken into sections called packets, which are routed individually through the network (see Figure 2.6). At the receiving device, the packets are reassembled to construct the complete message. Messages are divided into packets to ensure that large messages do not monopolize the network. Packets from several messages can be multiplexed through the same communication channel. Thus, packet switching enables devices to share the total network bandwidth efficiently.

Two variations of packet switching may be employed:

- Datagram services treat each packet as an independent message. The packets, also called datagrams, are routed through the network using the most efficient route currently available, enabling the switches to bypass busy segments and use underutilized segments. Datagrams frequently are employed on LANs and network layer protocols are responsible for routing the datagrams to the appropriate destination. Datagram service is called unreliable, not because it is inherently flawed but because it does not guarantee delivery of data. Recovery of errors is left to upper-layer protocols. Also, if several messages are required to construct a complete message, upper-layer protocols are responsible for reassembling the datagrams in order. Protocols that provide datagram service are called connectionless protocols.

- Virtual circuits establish a formal connection between two devices, giving the appearance of a dedicated circuit between the devices. When the connection is established, issues such as messages size, buffer capacities, and network paths are considered and mutually agreeable communication parameters are selected. A virtual circuit defines a connection, a communication path through the network, and remains in effect as the devices remain in communication. This path functions as a logical connection between the devices. When communication is over, a formal procedure releases the virtual circuit. Because virtual circuit service guarantees delivery of data, it provides reliable delivery service. Upper-layer protocols need not be concerned with error detection and recovery. Protocols associated with virtual circuits are called connection-oriented.

Chapter 3

Network Protocols

Protocols specify the services the network provides. A protocol is the specific set of rules that specify the meaning of messages exchanged by peer entities.

3.1 Chapter Structure

3.1.1 Outcomes

1. Demonstrate knowledge and understanding of the principles applicable to computer networking, particularly protocol design techniques and protocol layering.
2. Be able to construct, analyze and critically comment on network protocols and protocol design issues.
3. Evaluate the operation of various strategies and algorithms for achieving particular communication goals.
4. Appreciate the implication of existing network standardization issues (particularly adoption of the OSI standard), including the social issues involved.
5. Understand issues relating to communicating ideas, and how they correspond to communications protocols.
6. Be able to classify the protocol issues according to the OSI layering framework.
7. Suggest appropriate uses for standard network devices, and appreciate their position in terms of the OSI classification.

3.1.2 Objectives

This chapter:

1. Describes a range of issues that need to be addressed when designing and implementing communication protocols.
2. Details some standard solutions used to address each of these issues.
3. Views the communication problem as a set of solutions to each of the issues, and to decompose it vertically into layers.
4. Presents the OSI layering model, and discusses the implications of the standardization of such a model.
5. Introduces common network devices, and relates the operation of each to protocol issues occurring at specific levels in the OSI model.

3.2 Protocol Design and Specification

Building networks is a complex problem. Divide-and-conquer breaks a problem into smaller problems (subroutines or objects) which can be solved individually, and combined into a final solution. Networking extends this concept into layering. Starting with the hardware, we build successive layers on top of the lower layer. The idea is to build increasingly sophisticated services, using only the services provided by the layer immediately underneath.

Conceptually, layer *N* communicates with its remote peer (counterpart layer) on a remote machine. In reality, only the lowest layer physically transmits data. Each layer defines a protocol that describes the messages exchanged with its peer on the remote machine.

A crucial point in achieving good layering is abstraction - providing a well-defined service. When using the services of layer *N*, we are not concerned with how the service is actually implemented. We only need to know how to invoke it. Thus, the service better be one that is easy to use and allows the layer above it to build a better service.

Each layer provides an interface that specifies how its services are accessed by the layers above (and below) it. The interface should never change - changing it affects the layer above it. The actual implementation of the service, however, can change.

A network protocol specifies a method for ensuring effective communication. One or more of the issues below (see section 3.2.1) must be addressed by the design and specification of the protocol. Protocols which don't address all these issues will be used in a protocol stack, with a number of other protocols which collectively cover all the aspects of network communication.

3.2.1 Communication issues

3.2.1.1 Error Checking

A protocol needs to ensure integrity of delivery of a block of data from one computer to another. Error checking is used to ensure that data values arrive unchanged.

Different protocols may use different techniques to ensure reliable transfer of data. Common techniques employed include use of parity bits, checksums and CRCs (Cyclic Redundancy Codes) to ensure data is unchanged.

3.2.1.2 Sequencing

Packets may be delivered out of order - especially in systems that include multiple networks. Out of order delivery can be detected and corrected through sequencing. The sender attaches a sequence number to each outgoing packet. The receiver uses sequence numbers to put packets in order and detect missing packets.

3.2.1.3 Lost Packets

A common problem is lost packets. Any error, such as a bit error or incorrect length causes receiver to discard packet. This is a tough problem to solve. Although the sequence number can sometimes indicates gaps in the sequence, a problem arises when the missing packet is the last in the sequence, or if the sender requires a response before sending the next packet. The receiver needs other ways to decide when a packet has been lost.

Protocols can use positive acknowledgment with retransmission to detect and correct lost packets. The receiver always sends a short message acknowledging receipt of packets. The sender infers lost packets from missing acknowledgments and retransmits lost packets.

In practice, the sender sets timer for each outgoing packet and saves copy of packet. If the timer expires before acknowledgment is received, sender can retransmit the saved copy. The protocol must define upper bound on retransmission to detect unrecoverable network failure.

3.2.1.4 Duplicate Packets

Packets may be duplicated during transmission, or resent if an acknowledgment is delayed. The sequence number can also be used to detect duplicate packets with duplicated sequence numbers, and discard the duplicates.

Sufficiently delayed packets may end up inserted into later sessions. Suppose two computers exchange data with packets numbered 1 to 5. Packet 4 encounters an extraordinary delay; computers use retransmission to deliver valid copy of packet 4. The two computers exchange data later on with packets numbered 1 to 10. Initial **packet 4** can arrive during second session, so that the data from that old packet (rather than the current **packet 4**) is inserted into the data. Protocols may attach session number to each packet in a protocol session to differentiate packets from different sessions.

3.2.1.5 Flow control

Data overrun can occur when sender transmits data faster than receiver can process incoming data. Protocols use flow control mechanisms through which receiver can control rate of data transmission. These mechanisms include:

Stop-and-go The receiver sends small control packet when it is ready for next packet. The sender must wait for this control packet before sending next packet. Stop-and-go can be very inefficient with respect to network bandwidth if the time taken to transfer a packet between the two machines is large.

Sliding window This allows the sender to transmit multiple packets before receiving an acknowledgment. The number of packets that can be sent is defined by the protocol and called the window. As acknowledgments arrive from the receiver, the window is moved along the data packets; hence *sliding window*.

Credit scheme The receiver is able to issue credits to the sender. For every credit the sender receives, it may send a packet.

3.2.1.6 Addressing

The sender needs to be able to identify the sender, and vice versa. Some form of addressing is required to achieve this. Many protocols use the machine name, or number as part of the addressing scheme. This alone allows only one instance of the protocol to be used on a particular machine at a time. To refine this, many protocols use the concept of a port - which addresses a particular instance of the protocol on a single machine. For example, the TCP protocol uses port 23 to address the telnet server which uses TCP, port 21 for the FTP server which also uses TCP and port 80 for the HTTP (Web) server again using TCP. The combination of host machine address and port number uniquely identifies an instance of the protocol.

3.2.1.7 Connection Establishment

For datagram-oriented protocols, opening a connection simply allocates and initializes data structures in the operating system kernel. Connection oriented protocols often exchange messages that negotiate options with the remote peer at the time a connection is opened. Establishing a connection may be tricky because of the possibility of old or duplicate packets.

Finally, although not as difficult as establishing a connection, terminating a connection presents subtleties too. For instance, both ends of the connection must be sure that all the data in their queues have been delivered to the remote application.

TCP uses a 3-way handshake to initiate a connection. The handshake serves two functions:

1. It ensures that both sides are ready to transmit data, and that both ends know that the other end is ready before transmission actually starts.

2. It allows both sides to pick the initial sequence number to use.

An initial sequence number of 0 when opening a new connection is not suitable because if connections are of short duration, exchanging only a small number of segments, we may reuse low sequence numbers too quickly. Thus, each side that wants to send data must be able to choose its initial sequence number.

The 3-way handshake proceeds as follows:

1. A picks an initial sequence number (A_SEQ) and sends a segment to B containing: SYN=1, ACK=0, and SEQ=A_SEQ.
2. When B receives the SYN, it chooses its initial sequence number (B_SEQ) and sends a TCP segment to A containing: ACK=(A_SEQ+1), SEQ=B_SEQ, SYN=1.
3. When A receives B's response, it acknowledges B's choice of an initial sequence number by sending a dataless third segment containing: SYN=0, ACK=(B_SEQ+1), SEQ=A_SEQ+1.
4. Data transfer may now begin.

Note: The sequence number used in SYN segments are actually part of the sequence number space. That is why the third segment that A sends contains SEQ=(A_SEQ+1). This is required so that we don't get confused by old SYNs that we have already seen.

To insure that old segments are ignored, TCP ignores any segments that refer to a sequence number outside of its receive window. This includes segments with the SYN bit set.

3.2.1.8 Connection Termination

The Two-army problem is a classical analogy of the problem of terminating a communication when using unreliable messengers. Imagine a valley containing the White Army. Half the Blue Army A is above, on the northern slope of the valley; the other half, Blue Army B, is on the southern slope. The two halves of the Blue Army can only communicate by sending messengers across the valley, where the White Army might capture them. The White Army is big enough to win any battle against half a Blue Army; but would be overwhelmed if both halves of the Blues attacked simultaneously.

Suppose Blue A sends a message to Blue B saying "Do you agree that we attack at dawn?". Blue A will not attack unless it is known that the message got through and is accepted, so Blue B sends back a "Yes". Unfortunately Blue B does not know if the "Yes" got through so will not attack unless Blue A sends back "OK to your Yes". Unfortunately Blue A does not know if the "'OK to your Yes" got through...

In fact, the two armies cannot guarantee to synchronize because the communication channel is unreliable. For the same reason, we cannot guarantee to release a call without loss of information. Luckily this does not mean we cannot do so with low risk. In our army example, if we get as far as the first "OK to your Yes" then the risk of getting it wrong is far lower than the likely benefit. We reduce the risk by using a three-way handshake. A asks to disconnect and starts a timeout; B agrees and starts a timeout; A acknowledges and disconnects; B sees the acknowledgment and disconnects.

3.2.2 Protocol Implementation

3.2.2.1 Encapsulation

The software at each layer communicates with the corresponding layer through information stored in headers. Each layer adds its header to the front of the message from the next higher layer. Headers are nested at the front of the message as the message traverses the network.

On the sender, each layer:

- Accepts an outgoing message from the layer above.
- Adds a header and performs other processing.
- Passes resulting message to next lower layer.

On the receiver, each layer:

- Receives an incoming message from the layer below.
- Removes the header for that layer and performs other processing.
- Passes the resulting message to the next higher layer

3.2.2.2 Header Definition

Headers at each protocol layer can be defined abstractly. Ultimately however, they are intended to be communicated to lower protocol layers and ultimately transmitted over a network. For instance, a service at one layer may require transfer of certain abstract objects between computers; a lower layer may provide transfer services for strings of ones and zeroes, using encoding rules to transform the abstract objects into such strings.

One particular protocol may use a header such as:

```
struct header
  {
    byte source_address     [6];
    byte destination_address      [6];
    int length;
  }
```

This structure must be specified in a manner which is both independent of a particular host type (which may store the most significant byte first or last), and compiler which may use 2, 4 or 8 bytes to represent a primitive type such as an integer. The translation process from the abstract record definition to a sequence of octets (bytes or 8 bit values) allows the data to be correctly interpreted regardless of the nature of the machine connected to the network. Translation can occur using octet arrays (for fixed sized structures), or ASN.1 for more complex and versatile records.

3.2.2.2.1 Octet arrays A fixed arrays of octets is allocated, and the correspondence between each octet and the component of a field in the original record is specified. For example, the header used in section 3.2.2.2 could be translated as follows:

Octet	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Field	Src[0]	..				Src[5]	Dst[0]	..				Dst[5]	Length(MSB)	Length(LSB)

The majority of the protocols specified in this document use an octet array to specify their headers. The arrays are always linear, although are sometimes arranged slightly differently due to the finite width of the page. Some primitive types occupy more than one octet (such as integer length fields). The convention is always to use network byte order, which has the most significant bytes appear first within the octet array.

3.2.2.2.2 Java serialization Java has a useful facility, termed serialization, which can be used to transform objects into arrays of bytes (effectively octet arrays), and back again. This is very convenient for converting objects from a format convenient for use within a program, to a format suitable for transmission over a network.

A number of objects can be written to a single byte array using code such as that shown below. Note exception handling must still be dealt with:

```
// 3 fields to transmit over the network.
int linecount = 37;
String filename = "xyz.blend";
char p = 'p';
// these fields are arbitrary, don't try to
```

```
// read too much meaning into them.
// now add them to a byte stream, and eventually
// create a byte array from this.
ByteArrayOutputStream byteoutStream =
  new ByteArrayOutputStream ();
ObjectOutputStream objectStream =
  new ObjectOutputStream (byteoutStream);
objectStream.writeInt (linecount);
objectStream.writeObject (filename);
objectStream.writeChar (p);
objectStream.flush ();
byte[] bytedata = byteoutStream.toByteArray ();
```

Note the use of the writeObject method for objects that support serialization. This offers a way to convert whole classes in a single line.

The order in which objects are serialized and deserialized is, of course, very important. Java serialization stores not only the values of the objects, but extra information about the type of the objects. This allows it to detect if the stream is corrupted, or if there is an attempt to read the wrong type of object at any point.

The code to convert the byte array back to the original objects is very similar.

```
ByteArrayInputStream byteinStream =
  new ByteArrayInputStream (bytedata);
int l;
String s;
char c;
ObjectInputStream inputStream =
  new ObjectInputStream (byteinStream);
l = inputStream.readInt ();
s = (String) inputStream.readObject ();
c = inputStream.readChar ();
```

Serialization facilities are also available as additional libraries for a number of other languages.

3.2.2.2.3 Abstract Syntax Notation One Abstract objects can be specified in ASN.1 (Abstract Syntax Notation One, defined in X.208), and one set of rules for representing such objects as strings of ones and zeros is called the BER (Basic Encoding Rules, defined in X.209). This is a generic form of serialization which is independent of the language in which the ASN.1 and BER libraries are written.

ASN.1 is a flexible notation that allows one to define a variety data types, from simple types such as integers and bit strings to structured types such as sets and sequences, as well as complex types defined in terms of others. BER describes how to represent or encode values of each ASN.1 type as a string of eight-bit octets.

ASN.1 has four kinds of type:

1. Simple types, which are "atomic" and have no components. These include BIT STRING, an arbitrary string of bits (ones and zeroes), IA5String, an arbitrary string of IA5 (ASCII) characters, INTEGER, an arbitrary integer, NULL, a null value, OBJECT IDENTIFIER, an object identifier, which is a sequence of integer components that identify an object such as an algorithm or attribute type, OCTET STRING, an arbitrary string of octets (eight-bit values), PrintableString, an arbitrary string of printable characters, T61String, an arbitrary string of T.61 (eight-bit) characters and UTCTime, a "coordinated universal time" or Greenwich Mean Time (GMT) value.

2. Structured types, which have components. These are SEQUENCE, an ordered collection of one or more types, SEQUENCE OF, an ordered collection of zero or more occurrences of a given type, SET, an unordered collection of one or more types, SET OF, an unordered collection of zero or more occurrences of a given type.

3. Tagged types, which are derived from other types
4. Other types, which include the CHOICE type and the ANY type. The CHOICE type denotes a union of one or more alternatives; the ANY type denotes an arbitrary value of an arbitrary type, where the arbitrary type is possibly defined in the registration of an object identifier or integer value.

Every ASN.1 type other than CHOICE and ANY has a tag, which consists of a class and a nonnegative tag number. ASN.1 types are abstractly the same if and only if their tag numbers are the same. In other words, the name of an ASN.1 type does not affect its abstract meaning, only the tag does. Table 3.1 lists some ASN.1 types and their universal-class tags. ASN.1 types and values are expressed in a flexible,

Type Tag number	Tag number (decimal)	(hexadecimal)
INTEGER	2	02
BIT STRING	3	03
OCTET STRING	4	04
NULL	5	05
OBJECT IDENTIFIER	6	06
SEQUENCE	16	10
SET	17	11
Printable String	19	13
T61String	20	14
IA5String	22	16
UTCTime	23	17

Table 3.1: ASN.1 types and tag values.

programming-language-like notation.

3.2.2.2.4 Basic Encoding Rules The Basic Encoding Rules for ASN.1, abbreviated BER, give the way to represent any ASN.1 value as an octet string. The BER encoding has three or four parts:

- Identifier octets. These identify the class and tag number of the ASN.1 value, and indicate whether the method is primitive or constructed.
- Length octets. These give the number of contents octets.
- Contents octets. These give a concrete representation of the value, or the concatenation of the BER encodings of the components of the value.
- End-of-contents octets. For indefinite- length objects, these denote the end of the contents.

Simple types are encoded by

- Identifier octets. There are two forms: low tag number (for tag numbers between 0 and 30) and high tag number (for tag numbers 31 and greater).
    1. Low-tag-number form. One octet. Bits 8 and 7 specify the class, bit 6 has value "0," indicating that the encoding is primitive, and bits 5-1 give the tag number.
    2. High-tag-number form. Two or more octets. First octet is as in low-tag-number form, except that bits 5-1 all have value "1." Second and following octets give the tag number, base 128, most significant digit first, with as few digits as possible, and with the bit 8 of each octet except the last set to "1."
- Length octets. There are two forms: short (for lengths between 0 and 127), and long definite (for lengths between 0 and 21008-1).

1. Short form. One octet. Bit 8 has value "0" and bits 7-1 give the length.
2. Long form. Two to 127 octets. Bit 8 of first octet has value "1" and bits 7-1 give the number of additional length octets. Second and following octets give the length, base 256, most significant digit first.

- Contents octets. These give a concrete representation of the value.

Simple string types and structured types are BER encoded as follows:

- Identifier octets. As before, except that bit 6 has value "1," indicating that the encoding is constructed.

  Length octets. As before.

  Contents octets. The concatenation of the BER encodings of the components of the value.

3.3 The OSI stack

An architectural model developed by the International Standards Organization (ISO) is frequently used to describe the structure and function of data communication protocols. This architectural model, called the Open Systems Interconnect (OSI) Reference Model, contains seven layers that define the functions of data communications protocols. Each layer represents a function performed when data is transferred between cooperating applications across an intervening network. A layer does not define a single protocol, it defines a data communications function that may be performed by any number of protocols. Therefore, each layer may contain multiple protocols, each providing a service suitable to the function of that layer. Every protocol communicates with its peer. A peer is an implementation of the same protocol in the equivalent layer on a remote system. Each protocol is only concerned with communicating to its peer, it does not care about the layer above or below it. However, there must also be agreement on how to pass data between the layers on a single computer, because every layer is involved in sending data from a local application to an equivalent remote application. The individual layers do not need to know how the layers above and below them function, they only need to know how to pass data to them. Isolating network communications functions in different layers minimizes the impact of technological change on the entire protocol suite. New applications can be added without changing the physical network, and new network hardware can be installed without rewriting the application software. Although the OSI model is useful, the TCP/IP protocols don't match its structure exactly.

The approach used to designing a communication system is known as a layered architecture. Each layer has specific responsibilities and specific rules for carrying out those responsibilities, and knows nothing about the procedures the other layers follow. The layer carries out its task and delivers the message to the next layer in the process, and that is enough.

3.3.1 Characteristics of Layered Architectures

- They break the communication process into manageable chunks. Designing a small part of a process is much easier than designing the entire process, and simplifies engineering.
- A change at one layer does not affect the other layers. New delivery technology's can be introduced without affecting other layers.
- When a layer receives a message from an upper layer, the lower layer frequently encloses the message in a distinct package.
- The protocols at the various layers have the appearance of a stack, and a complete model of a data communication architecture is often called a protocol stack.
- Layers can be mixed and matched to achieve different requirements.
- Layers follow specific procedures for communicating with adjacent layers. The interfaces between layers must be clearly defined.

- An address mechanism is the common element that allows packets to be routed through the various layers until it reaches its destination. Sometimes, layers add their own address information.
- Essentially, each layer at the sender's end communicates with the corresponding layer at the receiver's end.
- Errors can occur at any of the layers. For critical messages, error-detecting mechanisms should be in place to either correct errors or notify the sender when they occur.

3.3.2 The Layers of the OSI model

The Open Systems Interconnect (OSI) reference model is the ISO (International Standards Organization) structure for the "ideal" network architecture. This Model outlines seven areas, or layers, for the network. These layers are (from highest to lowest):

7: Application: Where the user applications software lies. Such issues as file access and transfer, virtual terminal emulation, interprocess communication and the like are handled here.

6: Presentation: Differences in data representation are dealt with at this level. For example, UNIX-style line endings (CR only) might be converted to MS-DOS style (CRLF), or EBCIDIC to ASCII character sets.

5: Session: Communications between applications across a network is controlled at the session layer. Testing for out-of-sequence packets and handling two-way communication are handled here.

4: Transport: Makes sure the lower three layers are doing their job correctly, and provides a transparent, logical data stream between the end user and the network service s/he is using. This is the lower layer that provides local user services.

3: Network: This layer makes certain that a packet sent from one device to another actually gets there in a reasonable period of time. Routing and flow control are performed here. This is the lowest layer of the OSI model that can remain ignorant of the physical network.

2: Data Link: This layer deals with getting data packets on and off the wire, error detection and correction and retransmission. This layer is generally broken into two sub-layers: The LLC (Logical Link Control) on the upper half, which does the error checking, and the MAC (Medium Access Control) on the lower half, which deals with getting the data on and off the wire.
The LLC (Logical Link Control) is IEEE 802.2 Standard, and the MAC (Media Access Control) Layer is one of IEEE 802.3, .4, .5, .12, etc. standards or protocols.
In LANs, since you have several protocols defined in IEEE standards, it was decided to have a standard means of communicating with the Network Layer, but with the ability to communicate with different protocols communicating with the Data Link layer from below. The solution was to divide the Data Link layer into the layers LLC and MAC. The LLC provides services the the Network Layer, and uses the MAC layer to format frames and transmit frames using different protocols such as CSMA/CD (IEEE 802.3) or Token Ring (IEEE 802.5).

1: Physical: The nuts and bolts layer. Here is where the cable, connector and signalling specifications are defined.

There is also the undocumented but widely recognized ninth network layer: (Do not use in an exam!)

9: Bozone (a.k.a., loose nut behind the wheel): The user sitting at and using (or abusing, as the case may be) the networked device. All the error detection/correction algorithms in the world cannot protect your network from the problems initiated at the Bozone layer.

3.3.3 Advantages of the ISO OSI Model

- If a network conforms broadly to agreed standards, users are insulated against some of the adverse effects of technological change - equipment does not become obsolete quickly.
- It promotes modularization of network support software. Each module takes the form of a layer in the model and is responsible for providing selected services to the layer above.
- In theory any layer can be replaced by a new layer that provides the same services but in a different way, without affecting the user's view of the framework

3.3.4 Disadvantages of the ISO OSI Model

- Due to the complexity of the system poor performance is obtained, especially in some real time applications.
- Direct substitution of layers is not always possible e.g. if a LAN with broadcast capability is inserted below a network protocol that did not support this facility, then this service would be lost to the upper layers.
- Although protecting equipment from becoming obsolete it simultaneously hinders technological advancement.

3.4 Network Hardware

Data can be routed through an internetwork using the following three types of information:

- The physical address of the destination device, found at the data link layer. Devices that forward messages based on physical addresses generally are called bridges.
- The address of the destination network, found at the network layer. Devices that use network addresses to forward messages usually are called routers, although the original name, still commonly used in the TCP/IP world, is gateway.
- The circuit that has been established for a particular connection. Devices that route messages based on assigned circuits are called switches.

3.4.1 Repeater

A repeater acts on a purely electrical level to connect to segments. All it does is amplify and reshape (and, depending on the type, possibly re-time) the analog waveform to extend network segment distances. It does not know anything about addresses or forwarding, thus it cannot be used to reduce traffic as a bridge can in the example above.

3.4.2 Hub

A hub is a common wiring point for star-topology networks, and is a common synonym for concentrator (though the latter generally has additional features or capabilities). Arcnet, 10Base-T Ethernet and 10Base-F Ethernet and many proprietary network topologies use hubs to connect multiple cable runs in a star-wired network topology into a single network. Token-Ring MSAUs (Multi-Station Access Units) can also be considered a type of hub, but don't let a token-ring bigot hear that. Hubs have multiple ports to attach the different cable runs. Some hubs (such as 10Base-T and active Arcnet) include electronics to regenerate and re-time the signal between each hub port. Others (such as 10Base-F or passive Arcnet) simply act as signal splitters, similar to the multi-tap cable-TV splitters you might use on your home antenna coax (of course, 10Base-F uses mirrors to split the signals between cables). Token- Ring MSAUs use relays (mechanical or electronic) to reroute the network signals to each active device in series, while all other hubs redistribute received signals out all ports simultaneously, just as a 10Base2 multi-port repeater would.

3.4.3 Bridge

A bridge will connect to distinct segments (usually referring to a physical length of wire) and transmit traffic between them. This allows you to extend the maximum size of the network while still not breaking the maximum wire length, attached device count, or number of repeaters for a network segment.

A bridge must implement both the physical and data link layers of the protocol stack. Bridges are fairly simple devices. They receive frames from one connection and forward them to another connection known to be en route to the destination. When more than one route is possible, bridges ordinarily can't determine which route is most efficient. In fact, when multiple routes are available, bridging can result in frames simply traveling in circles. Having multiple paths available on the network is desirable, however, so that a failure of one path does not stop the network. With Ethernet, a technique called the spanning-tree algorithm enables bridged networks to contain redundant paths.

Token Ring uses a different approach to bridging. When a device needs to send to another device, it goes through a discovery process to determine a route to the destination. The routing information is stored in each frame transmitted and is used by bridges to forward the frames to the appropriate networks. Although this actually is a data link layer function, the technique Token Ring uses is called source routing.

The bridge must implement two protocol stacks, one for each connection. Theoretically, these stacks could belong to different protocols, enabling a bridge to connect different types of networks. However, each type of network, such as Ethernet and Token Ring, has its own protocols at the data link layer. Translating data from the data link layer of an Ethernet to the data link layer of a Token Ring is difficult, but not impossible. Bridges, which operate at the data link layer, therefore, generally can join only networks of the same type. You see bridges employed most often in networks that are all Ethernet or all Token Ring. A few bridges have been marketed that can bridge networks that have different data link layers.

3.4.3.1 Learning Bridge

A learning bridge monitors MAC (OSI layer 2) addresses on both sides of its connection and attempts to learn which addresses are on which side. It can then decide when it receives a packet whether it should cross the bridge or stay local (some packets may not need to cross the bridge because the source and destination addresses are both on one side). If the bridge receives a packet that it doesn't know the addresses of, it will forward it by default.

3.4.3.2 Remote Bridge

A bridge as described above that has an Ethernet interface on one side and a serial interface on the other. It would connect to a similar device on the other side of the serial line. Most commonly used in WAN links where it is impossible or impractical to install network cables. A high-speed modem (or T1 DSU/CSUs, X.25 PADs, etc) and intervening telephone lines or public data network would be used to connect the two remote bridges together.

3.4.4 Routers

Routers work much like bridges, but they pay attention to the upper network layer protocols (OSI layer 3) rather than physical and data link layer (OSI layer 1 & 2) protocols. A router will decide whether to forward a packet by looking at the protocol level addresses (for instance, TCP/IP addresses) rather than the MAC address. Because routers work at layer 3 of the OSI stack, it is possible for them to transfer packets between different media types (i.e., leased lines, Ethernet, token ring, X.25, Frame Relay and FDDI). Many routers can also function as bridges.

A different method of path determination can be employed using data found at the network layer. At that layer, networks are identified by logical network identifiers. This information can be used to build a picture of the network. This picture can be used to improve the efficiency of the paths that are chosen. Devices that forward data units based on network addresses are called routers.

With TCP/IP, routing is a function of the network layer. By convention, the network on which the data unit originates counts as one hop. Each time a data unit crosses a router, the hop count increases by one. This assumes that all of the paths between the routers provide the same rate of service. A simple hop-count

algorithm would be misleading if some lines have greater capacity than others. Apart from such extreme cases, however, hop-count routing is a definite improvement over no routing planning at all.

Routing operates at the network layer. By the time data reach that layer, all evidence of the physical network has been removed. Both protocol stacks in the router can share a common network layer protocol. The network layer does not know or care if the network is Ethernet or Token Ring. Therefore, each stack can support different data link and physical layers. Consequently, routers posses a capability, fairly rare in bridges, to forward traffic between dissimilar types of networks. Owing to that capability, routers often are used to connect LANs to WANs.

Building routers around the same protocol stack as are used on the end-nodes is possible. TCP/IP networks can use routers based on the same IP protocol employed at the workstation. However, it is not required that routers and end-nodes use the same routing protocol. Because network layers need not communicate with upper-layer protocols, different protocols may be used in routers than are used in the end-nodes. Commercial routers employ proprietary network layer protocols to perform routing. These custom protocols are among the keys to the improved routing performance provided by the bets routers.

3.4.5 Bridges versus Routers

When deciding whether to use a bridge or router in a particular situation your network layout, type and amount of hosts and traffic, and other issues (both technical and non-technical) must be considered. Routing would always be preferable to bridging except that routers are slower and usually more expensive (due to the amount of processing required to look inside the physical packet and determine which interface that packet needs to get sent out), and that many applications use non-routeable protocols (i.e., NetBIOS, DEC LAT, etc.).

Rules of thumb:

- Bridges are usually good choices for small networks with few, if any, slow redundant links between destinations. Further, bridges may be your _only_ choice for certain protocols, unless you have the means to encapsulate (tunnel) the unrouteable protocol inside a routeable protocol.
- Routers are usually much better choices for larger networks, particularly where you want to have a relatively clean WAN backbone. Routers are better at protecting against protocol errors (such as broadcast storms) and bandwidth utilization. Since routers look deeper inside the data packet, they can also make forwarding decisions based on the upper-layer protocols.

Occasionally, a combination of the two devices are the best way to go. Bridges can be used to segment small networks that are geographically close to each other, between each other and the router to the rest of the WAN.

3.4.6 Switches

On a shared Ethernet network, data is sent to a hub which then rebroadcasts this data to all ports on the network until it gets to its proper destination. When more users are added to the network, more data signals are broadcast, and consequently, more signals can collide with each other, causing the network to slow down.

Switched Ethernet, on the other hand, provides dedicated bandwidth through a "private" connection between two devices on a network. There are no collisions because every user is essentially on a private line. Switched Ethernet, which is based on standard Ethernet and uses the same wiring types, is able to detect the destination of data being sent along the network, and forwards that data directly to the place it's going rather than rebroadcasting it to every port on the hub. Switched Ethernet is like having your own private phone line instead of sharing phone privileges on a party line.

In the most basic type of network found today, nodes are simply connected together using hubs. As a network grows, there are some potential problems with this configuration:

- Scalability: In a hub network, limited shared bandwidth makes it difficult to accommodate significant growth without sacrificing performance. Applications today need more bandwidth than ever before. Quite often, the entire network must be redesigned periodically to accommodate growth.

- Latency: The amount of time that it takes a packet to get to its destination. Since each node in a hub-based network has to wait for an opportunity to transmit in order to avoid collisions, the latency can increase significantly as you add more nodes. Or if someone is transmitting a large file across the network, then all of the other nodes are waiting for an opportunity to send their own packets. You have probably seen this before at work. You try to access a server or the Internet and suddenly everything slows down to a crawl.

- Network Failure: In a typical network, one device on a hub can cause problems for other devices attached to the hub due to wrong speed settings (100Mbps on a 10Mbps hub) or excessive broadcasts. Switches can be configured to limit broadcast levels.

- Collisions: Ethernet uses a process called Carrier Sense Multiple Access with Collision Detection (CSMA/CD) to communicate across the network. Under CSMA/CD, a node will not send out a packet unless the network is clear of traffic. If two nodes send out packets at the same time, a collision occurs and the packets are lost. Then both nodes wait a random amount of time and retransmit the packets. Any part of the network where there is a possibility that packets from two or more nodes will interfere with each other is considered to be part of the same collision domain. A network with a large number of nodes on the same segment will often have a lot of collisions and therefore a large collision domain.

While hubs provide an easy way to scale up and shorten the distance that the packets must travel to get from one node to another, they do not break up the actual network into discrete segments. That is where switches come in.

Think of a hub as a four-way intersection where everyone has to stop. If more than one car reaches the intersection at the same time, they have to wait for their turn to proceed. But a switch is like a cloverleaf intersection. Each car can take an exit ramp to get to their destination without having to stop and wait for other traffic to go by. Now imagine what this would be like with a dozen or even a hundred roads intersecting at a single point. The amount of waiting and the potential for a collision increases significantly if every car has to check all the other roads before proceeding. But wouldn't it be amazing if you could take an exit ramp from any one of those roads to the road of your choosing? That is exactly what a switch does for network traffic!

3.4.6.1 Benefits of switches

A vital difference between a hub and a switch is that all the nodes connected to a hub share the bandwidth among themselves while a device connected to a switch port has the full bandwidth all to itself. For example, if 10 nodes are communicating using a hub on a 10 Mbps network, then each node may only get a portion of the 10 Mbps if other nodes on the hub want to communicate as well. But with a switch, each node could possibly communicate at the full 10 Mbps. Think about our road analogy. If all of the traffic is coming to a common intersection, then it has to share that intersection with everyone else. But a cloverleaf allows all of the traffic to continue at full speed from one road to the next.

In a fully switched network, switches replace all the hubs of an Ethernet network with a dedicated segment for every node. These segments connect to a switch, which supports multiple dedicated segments (sometimes in the hundreds). Since the only devices on each segment are the switch and the node, the switch picks up every transmission before it reaches another node. The switch then forwards the frame over the appropriate segment. Since any segment contains only a single node, the frame only reaches the intended recipient. This can allow many conversations to occur simultaneously on a switched network.

Switching allows a network to maintain full-duplex Ethernet. Before switching, Ethernet was half-duplex, which means that only one device on the network can transmit at any given time. In a fully switched network, nodes only communicate with the switch and never directly with each other. Using our road analogy, half-duplex is similar to the problem of a single lane, like when road construction closes down the use of one lane of a two lane road. Traffic is trying to use the same lane in both directions. This means that traffic coming one way must wait until traffic from the other direction stops. Otherwise, they will hit head-on!

Fully switched networks employ either twisted pair or fiber optic cabling, both of which use separate conductors for sending and receiving data. In this type of environment, Ethernet nodes can forgo the collision detection process and transmit at will, since they are the only potential devices that can access the medium. In other words, traffic flowing in each direction has a lane to itself. This allows nodes to transmit to the switch at the same time the switch transmits to them, achieving a collision free environment. Transmitting in both directions also can effectively double the apparent speed of the network when two nodes are exchanging information. For example, if the speed of the network is 10 Mbps then each node can transmit at 10Mbps at the same time.

Most networks are not fully switched because of the costs incurred in replacing all of the hubs with switches. Instead, a combination of switches and hubs are used to create an efficient yet cost-effective network. For example, a company may have hubs connecting the computers in each department and a switch connecting all of the department-level hubs together.

The use of switches has a number of advantages:

- Proven Technology: Because switching is based on 10BASE-T Ethernet, most people are very comfortable with it and are willing to integrate it into their existing network.
- Preserves Current Infrastructure: Switching uses the same cabling and network adapters as shared Ethernet. This makes it easy to connect a switch directly to any 10BASE-T device you already have on your network.
- Simplicity and Cost: the cost of switching products falling is rapidly, and switching doesn't require purchasing new network adapters or cabling.

Switches are well suited to situations where:

- several users on my network are competing for bandwidth on a fairly constant basis. If large files are being transmitted intermittently, you may want to look into Fast Ethernet's performance.
- older category 3 twisted-pair cabling is used for the existing shared Ethernet network. If your building is is newer and already wired for Category 5 twisted-pair cabling, Fast Ethernet may be a good choice.
- budget is limited, for upgrades, and network management overheads. Although they do require some network management skills, switches are fairly easy install and can be inexpensive.

Circuit-based networks operate with high efficiency because the path is established once, when the circuit is established. Each switch maintains a table that records how data from different circuits should be switched. Switching is typically performed by lower-level protocols to enhance efficiency, and is associated most closely with the data link layer.

3.4.6.2 Switching Technologies

You can see that a switch has the potential to radically change the way the nodes can communicate with each other. But you may be wondering what makes it different from a router. Switches usually work at Layer 2 (Data or Data Link) of the OSI Reference Model using MAC addresses while routers work at Layer 3 (Network) with Layer 3 addresses (IP, IPX, or Appletalk depending on what Layer 3 protocols are being used). The algorithm that switches use to decide how to forward packets is different from the algorithms used by routers to forward packets. One of these differences in the algorithms between switches and routers is how broadcasts are handled. On any network, the concept of a broadcast packet is vital to the operability of a network. Whenever a device needs to send out information but doesn't know who it should send it to, it sends out a broadcast. For example, every time a new computer or other device comes on to the network, it sends out a broadcast packet to announce its presence. The other nodes (such as a domain server) can add the computer to their browser list (kind of like an address directory) and communicate directly with that computer from that point on. Broadcasts are used any time a device needs to make an announcement to the rest of the network or is unsure of who the recipient of the information should be.

A hub or a switch will pass along any broadcast packets they receive to all the other segments in the broadcast domain but a router will not. Think about our four way intersection again. In our analogy, all of the traffic passed through the intersection no matter where it was going. Now imagine that this intersection is at an international border. To pass through the intersection, you must provide the border guard with the specific address that you are going to. If you don't have a specific destination, then the guard will not let you pass. A router works like this. Without the specific address of another device, it will not let the data packet through. This is a good thing for keeping networks separate from each other but not so good when you want to talk between different parts of the same network. This is where switches come in.

LAN switches rely on Packet-switching. The switch establishes a connection between two segments just long enough to send the current packet. Incoming packets (part of an Ethernet frame) are saved to a temporary memory area (buffer), the MAC address contained in the frame's header is read and then compared to a list of addresses maintained in the switch's lookup table . In an Ethernet-based LAN, an Ethernet frame contains a normal packet as the payload of the frame with a special header that includes the MAC address information for the source and destination of the packet.

Packet-based switches use one of three methods for routing traffic:

Cut-through switches: Cut-through switches read the MAC address as soon as a packet is detected by the switch. After storing the six bytes that make up the address information, they immediately begin sending the packet to the destination node, even though the rest of the packet is coming into the switch.

Store and forward: A switch using store and forward will save the entire packet to the buffer and check it for Cyclic Redundancy Check (CRC) errors or other problems. If the packet has an error, then it is discarded. Otherwise, the switch looks up the MAC address and sends the packet on to the destination node. Many switches combine the two methods by using cut-through until a certain error level is reached, then changing over to store and forward. Very few switches are strictly cut-through since this provides no error correction.

Fragment free: A less common method is fragment-free. It works like cut-through but stores the first 64 bytes of the packet before sending it on. The reason for this is that most errors and all collisions occur during the initial 64 bytes of a packet.

LAN switches vary in their physical design. Currently, there are three popular configurations in use:

Shared-memory - Stores all incoming packets in a common memory buffer shared by all the switch ports (input/output connections), then sends them out the correct port for the destination node.

Matrix - This type of switch has an internal grid with the input ports and the output ports crossing each other. When a packet is detected on an input port, the MAC address is compared to the lookup table to find the appropriate output port. The switch then makes a connection on the grid where these two ports intersect.

Bus-architecture - Instead of a grid, an internal transmission path (common bus) is shared by all of the ports using Time Division Multi Access (TDMA).

Most Ethernet LAN switches use a very cool system called transparent bridging to create their address lookup tables. Transparent bridging is a technology that allows a switch to learn everything it needs to know about the location of nodes on the network without the network administrator having to do anything.

When a switch gets the first packet of data from a node it reads the MAC address and saves it to the lookup table. The switch now knows where to find this node anytime a packet is addressed to it. This process is called learning.

When the switch does not know where a node is, it sends the packet to all of the segments except the one that it arrived on. This is called flooding.

If the switch determines that both source and destination nodes for a given packet are on the same segment, it will ignore packets traveling between nodes on the same segment. This is filtering.

Most switches have plenty of memory in a switch for maintaining the lookup tables, but remove older information so that the switch doesn't waste time searching through stale addresses. To optimize the use of this memory, switches use a technique called aging. Basically, when an entry is added to the lookup table for a node, it is given a timestamp. Each time a packet is received from a node, the timestamp is updated. The switch has a user-configurable timer that erases the entry after a certain length of time with no activity from that node. This frees up valuable memory resources for other entries.

3.4.6.3 Cyclic networks

If switch fails then the network will be segmented, or even brought down. Additional links between switches provides redundancy and effectively eliminates the single point of failure.

Now we have a new problem. With all of the switches now connected in a loop, a packet from a node could quite possibly come to a switch from two different segments. The flood packets sent when looking for a node will reach the original switch via a different port, triggering additional flooding packets. This causes a broadcast storm as the packets are broadcast, received and rebroadcast by each switch resulting in potentially severe network congestion.

To prevent broadcast storms and other unwanted side effects of looping, Digital Equipment Corporation created the Spanning Tree Protocol (STP) which has been standardized as the 802.1d specification by the Institute of Electrical and Electronic Engineers (IEEE). Essentially, a spanning tree uses the spanning tree algorithm (STA) which senses that the switch has more than one way to communicate with a node, determines which way is the best and blocks out the other path(s). The cool thing is that it keeps track of the other path(s) just in case the primary path is unavailable.

3.4.6.4 Routers and Layer 3 Switching

While most switches operate at the Data layer (Layer 2) of the OSI Reference Model, some incorporate features of a router and operate at the Network layer (Layer 3) also. In fact, a Layer 3 switch is incredibly similar to a router.

Like routers, Layer 3 switches actually work at the Network layer. When a router receives a packet, it looks at the Layer 3 (the Network Layer) source and destination addresses to determine the path the packet should take. This is considered Layer 3 (Network) networking activity. A standard switch relies on the MAC addresses to determine the source and destination of a packet, which is Layer 2 (Data) networking. The fundamental difference between a router and a Layer 3 switch is that Layer 3 switches have optimized hardware to pass data as fast as Layer 2 switches, yet they make decisions on how to transmit traffic at Layer 3, just like a router would. Within the LAN environment, a Layer 3 switch is usually faster than a router because it is built on switching hardware. In fact, many of Cisco's Layer 3 switches are actually routers that operate faster because they are built on "switching" hardware with customized chips inside the box.

3.4.6.5 VLANs

As networks have grown in size and complexity, many companies have turned to Virtual Local Area Networks (VLANs) to provide some way for structuring this growth logically. Basically, a VLAN is a collection of nodes that are grouped together in a single broadcast domain that is based on something other than physical location. A broadcast domain is a network (or portion of a network) that will receive a broadcast packet from any node located within that network. In a typical network, everything on the same side of the router is all part of the same broadcast domain. A switch that you have implemented VLANs on now has multiple broadcast domains similar to a router. But you still need a router to route from one VLAN to another, the switch can't do this by itself.

Here are some common reasons that a company might have VLANs:

- Security - Separating systems with sensitive data from the rest of the network decreases the chance that someone will gain access to information they are not authorized to see.

- Projects/Special applications - Managing a project or working with a specialized application can be simplified by the use of VLAN that brings all of the required nodes together.
- Performance/Bandwidth - Careful monitoring of network use allows the network administrator to create VLANs that reduce the number of router hops and increase the apparent bandwidth for network users.
- Broadcasts/Traffic flow - Since a principle element of a VLAN is the fact that it does not pass broadcast traffic to nodes that are not part of the VLAN, it automatically reduces broadcasts. Access lists provide the network administrator with a way to control who sees what network traffic. An access list is a table the network administrator creates that lists what addresses have access to that network.
- Departments/Specific job types - Companies may want VLANs set up for departments that are heavy network users (such as Multimedia or Engineering) or a VLAN across departments that is dedicated to specific types of employees (such as managers or sales people).

While you can have more than one VLAN on a switch, they cannot communicate directly with each other. If they did it would defeat the purpose of having a VLAN, which is to isolate a part of the network. To communicate between VLANs requires the use of a router.

VLANs can span across multiple switches and you can have more than one VLAN on each switch. For multiple VLANs on multiple switches to be able to communicate via a single link between the switches, you must use a process called trunking; trunking is the technology that allows information from multiple VLANs to be carried over just one link between switches.

Chapter 4

Network Signal Transmission: Physical Layer

4.1 Chapter Structure

4.1.1 Outcomes

1. Demonstrate knowledge and understanding of bit encoding, framing and error detection.

4.1.2 Objectives

This chapter:

1. Describes signalling techniques, including those used in the physical layer of many networks.

2. Describes framing and synchronization techniques used to ensure data gets on and off the wire correctly.

4.2 Introduction

The first step in turning nodes and links into usable building blocks is to understand how to connect them in such a way that bits can be transmitted from one node to the other. Signals propagate over physical links. The task, therefore, is to encode the binary data that the source node wants to send into the signals that the links are able to carry, and then decode the signal back into the corresponding binary data at the receiving node. We consider this problem in the context of a digital link, in which case we are most likely working with two discrete signals. We generically refer to these as the high signal and the low signal, although in practice these signals would be two different voltages on a copper-based link and two different power levels on an optical link.

The physical layer (layer 1) of the OSI reference model serializes the frame (i.e. converts it to a series of bits) and sends it across a communications circuit (i.e cable) to the destination (or an intermediate) system.

4.3 Signalling of Bits

The physical layer defines the representation of each bit as a voltage, current, phase, or frequency. A number of common schemes exist.

4.3.1 NRZ, Non Return to Zero transmission (Level signalling)

In NRZ transmission, each data bit is represented by a level. A high level may represent a logic 1, where as a low level may represent a logic 0. NRZ encoding is illustrated in Figure 4.1.

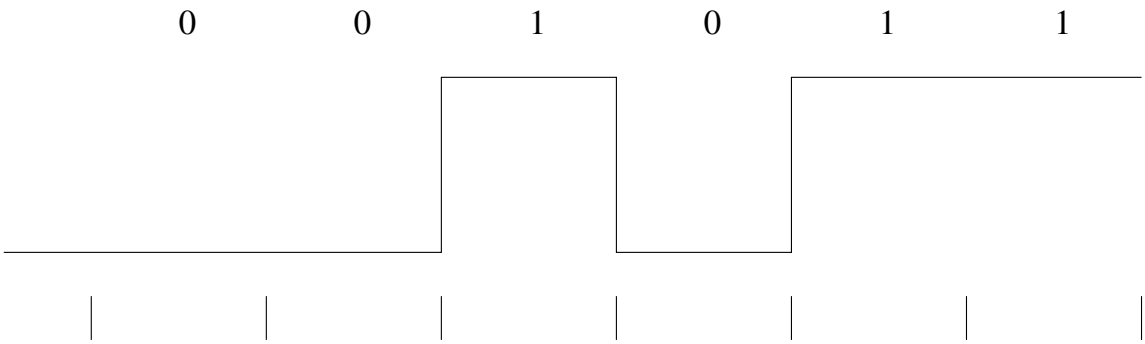


Figure 4.1: NRZ encoding

Non-return to zero encoding is commonly used in slow speed communications interfaces for both synchronous and asynchronous transmission. Using NRZ, a logic 1 bit is sent as a high value and a logic 0 bit is sent as a low value (the line driver chip used to connect the cable may subsequently invert these signals).

A problem arises when using NRZ to encode a synchronous link which may have long runs of consecutive bits with the same value. Figure 4.2 illustrates the problem that would arise if NRZ encoding were used. In Ethernet for example, there is no control over the number of 1's or 0's which may sent consecutively. There could potentially be thousands of 1's or 0's in sequence. If the encoded data contains long 'runs' of logic 1's or 0's, this does not result in any bit transitions. The lack of transitions prevents the receiver from reliably regenerating the clock making it impossible to detect the boundaries of the received bits at the receiver. This is the reason why Manchester coding is used in Ethernet LANs.

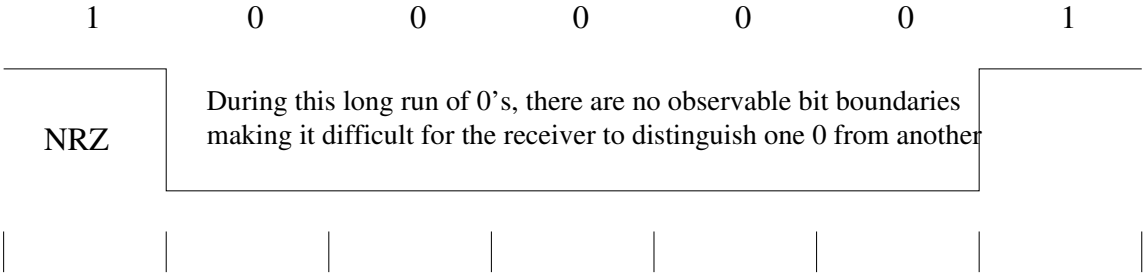


Figure 4.2: A long run of bits with the same value results in no transitions on the cable when NRZ encoding is used.

4.3.2 RZ, Return to Zero (Pulse signalling)

Pulses are used to represent bits (called Return to Zero, RZ) in which a logic 1 is represented by a pulse and a logic 0 by the absence of a pulse. Half way through the pulse, the signal returns to zero. This allows a clock signal to be extracted from the logic 1 signals, at the cost of halving the width of the pulses.

Polar Return to Zero using both positive and negative voltages, and sends pulses for both logic 1s and 0s. Thus the clock signal can be easily extracted for any bit, at the cost of an extra voltage level.

Unipolar and polar RZ encoding is illustrated in Figure 4.3.

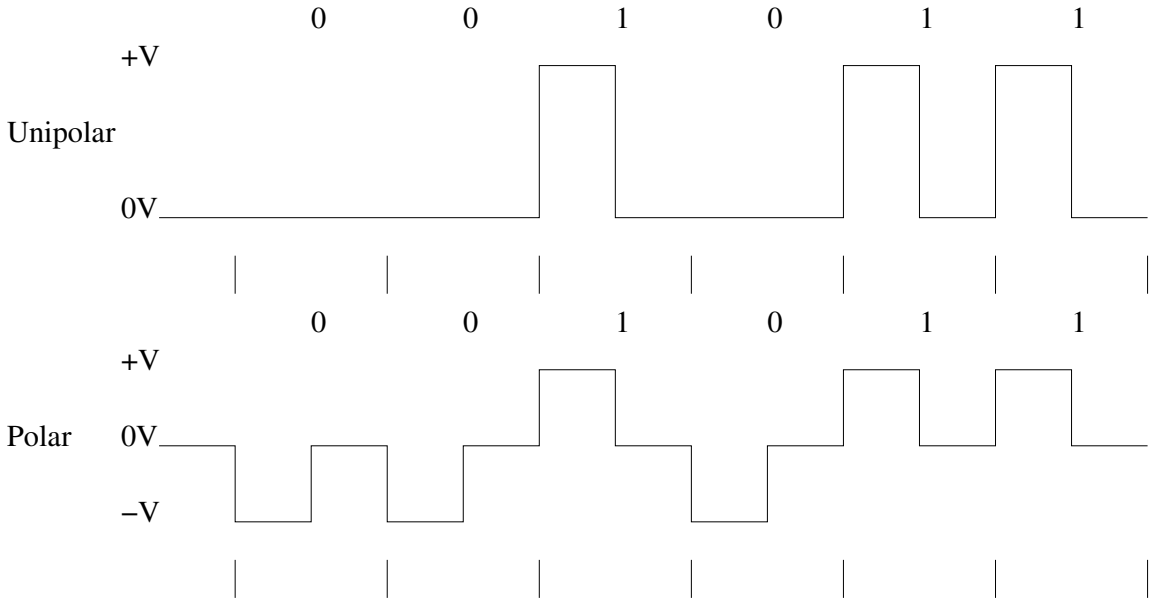


Figure 4.3: RZ Encoding (Unipolar and Polar)

4.3.3 Manchester encoding (Edge/phase signalling)

Manchester encoding uses a scheme where a logic 1 is represented by a transition in a particular direction (usually a rising edge) in the center of each bit. A transition in the opposite direction (downward in this case) is used to represent a logic 0.

Manchester encoding is a synchronous clock encoding technique used by the OSI physical layer to encode the clock and data of a synchronous bit stream. In this technique, the actual binary data to be transmitted over the cable are not sent as a sequence of logic 1's and 0's (known technically as Non Return to Zero (NRZ)). Instead, the bits are translated into a slightly different format that has a number of advantages over using straight binary encoding (i.e. NRZ).

Manchester encoding follows the rules shown below:

Original Data	Value Sent
Logic 0	1 to 0 (downward transition at bit center)
Logic 1	0 to 1 (upward transition at bit center)

The diagram in Figure 4.4 shows a typical Manchester encoded signal with the corresponding binary representation of the data (0,0,1,0,1,1) being sent.

In the Manchester encoding shown, a logic 1 is indicated by a 0 to 1 transition at the center of the bit and a logic 0 is indicated by a 1 to 0 transition at the center of the bit. Note that signal transitions do not always occur at the 'bit boundaries' (the division between one bit and another), but that there is always a transition at the center of each bit. (N.B. since most line driver electronics actually inverts the bits prior to transmission, you may observe the opposite coding on an oscilloscope connected to a cable).

A Manchester encoded signal contains frequent level transitions which allow the receiver to extract the clock signal using a Digital Phase Locked Loop (DPLL) and correctly decode the value and timing of each bit. To allow reliable operation using a DPLL, the transmitted bit stream must contain a high density of bit transitions. Manchester encoding ensures this, allowing the receiving DPLL to correctly extract the clock signal.

Example of Manchester encoding

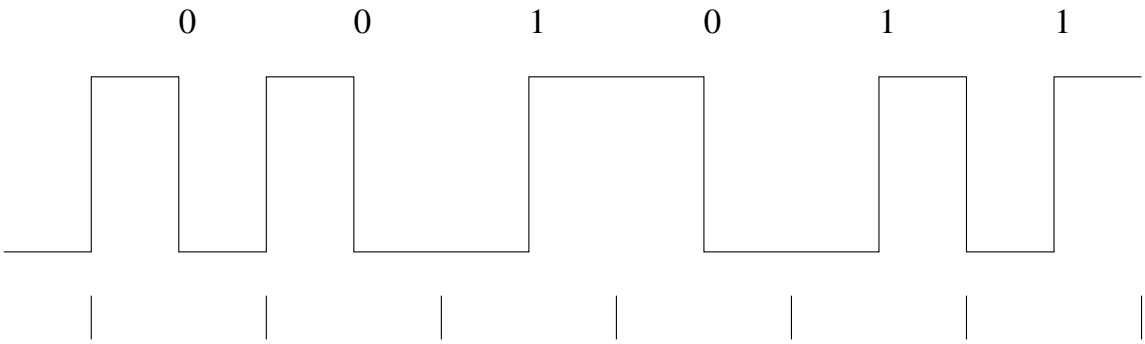


Figure 4.4: The waveform for a Manchester encoded bit stream carrying the sequence of bits 001011.

- The pattern of bits " 1 0 0 0 0 1 1 0 " encodes to " 01 10 10 10 10 01 01 10".
- Another more curious example is the pattern "0 1 0 1 0 etc" which encodes to "10 01 10 01 10 " which could also be viewed as "1 00 11 00 11 0 ". Thus for a 10 Mbps Ethernet LAN, the preamble sequence encodes to a 10 MHz square wave!

4.4 Timing of Bits

At the receiver, the remote system reassembles the series of bits to form a frame and forwards the frame for processing by the link layer. A clock (timing signal) is needed to identify the boundaries between the bits (in practice it is preferable to identify the center of the bit - since this usually indicates the point of maximum signal power). There are two systems used to providing timing:

4.4.1 Asynchronous Communication (independent transmit & receive clocks)

The asynchronous communication technique is a physical layer transmission technique which is most widely used for personal computers providing connectivity to printers, modems, fax machines, etc. The most significant aspect of asynchronous communications is that the transmitter and receiver clock are independent and are not synchronized. In fact, there need be no timing relationship between successive characters (or bytes of data). Individual characters may be separated by any arbitrary idle period.

An asynchronous link communicates data as a series of characters of fixed size and format. Each character is preceded by a start bit and followed by 1-2 stop bits. Parity is often added to provide some limited protection against errors occurring on the link. The use of independent transmit and receive clocks constrains transmission to relatively short characters (<8 bits) and moderate data rates (< 64 Kbps, but typically lower). The asynchronous transmitter delimits each character by a start sequence and a stop sequence. The start bit (0), data (usually 8 bits plus parity) and stop bit(s) (1) are transmitted using a shift register clocked at the nominal data rate.

At the receiver, a clock of the same nominal frequency is constructed and used to clock-in the data to the receive shift register. Only data that are bounded by the correct start and stop bits are accepted. This operation is normally performed using a UART - Universal Asynchronous Receiver Transmitter.

The reconstructed receive clock is normally generated using a local stable high rate clock, frequently operating at 16 or 32 times the intended data rate. Clock generation proceeds by detecting the edge of the start bit and counting sufficient clock cycle from the high frequency clock to identify the mid position of the start bit. From there the center of the successive bits are located by counting cycles corresponding to the original data speed.

When asynchronous transmission is used to support packet data links (e.g. IP), then special characters have to be used ("framing") to indicate the start and end of each frame transmitted. One character (known as an escape character) is reserved to mark any occurrence of the special characters within the frame. In

this way the receiver is able to identify which characters are part of the frame and which are part of the "framing".

4.4.2 Synchronous Communication (synchronized transmit & receive clocks)

The principle difference between the synchronous and asynchronous modes of transmission is that in the synchronous case, the receiver uses a clock which is synchronized to the transmitter clock. The clock may be transferred by:

- A separate interface circuit (as in X.21 or RS-449)
- Encoded in the data (e.g. Manchester Encoding, AMI encoding, HDB3 encoding)

Synchronous transmission has the advantage that the timing information is accurately aligned to the received data, allowing operation at much higher data rates. It also has the advantage that the receiver tracks any clock drift which may arise (for instance due to temperature variation). The penalty is however a more complex interface design, and potentially a more difficult interface to configure (since there are many more interface options).

4.5 Framing and Synchronization

Here we will consider aspects of transferring units of data larger than a single analog or digital encoding symbol. We will need to recover clock information for both the signal (so we can recover the right number of symbols and recover each symbol as accurately as possible), and obtain synchronization for larger units of data (such as data words and frames). It is necessary to recover the data in words or blocks because this is the only way the receiver process will be able to interpret the data received; for a given bit stream, depending on the byte boundaries there will be seven or eight ways to interpret the bit stream as ASCII characters, and these are likely to be very different.

In order to receive bits in the first place, the receiver must be able to determine how fast bits are being sent and when it has received a signal symbol (usually one bit, but with m-ary signalling, it may be several bits, log(m) in fact). Further, the receiver needs to be able to determine what the relationship of the bits in the received stream have to one another, that is, what the logical units of transfer are, and where each received bit fits into the logical units. We call these logical units frames. This means that in addition to bit (or transmission symbol) synchronization, the receiver needs word and frame synchronization.

4.5.1 Bit synchronization

In asynchronous communication, small, fixed-length words (usually 5 to 9 bits long) are transferred without any clock line or clock recovery from the signal itself. Each word has a start bit (a 0) before the first data bit of the word and a stop bit (a 1) after the last data bit of the word. The receiver's local clock is started when the receiver detects the 1-0 transition of the start bit, and the line is sampled in the center of the fixed bit intervals (a bit interval is the inverse of the data rate).

The sender outputs the bit at the agreed-upon rate, holding the line in the appropriate state for one bit interval for each bit, but using its own local clock to determine the length of these bit intervals. The receiver's clock and the sender's clock may not run at the same speed, so that there is a relative clock drift (this may be caused by variations in the crystals used, temperature, voltage, etc.).

If the receiver's clock drifts too much relative to the sender's clock, then the bits may be sampled while the line is in transition from one state to another, causing the receiver to misinterpret the received data.

Data must be sent in multiples of the data length of the word, and the two or more bits of synchronization overhead compared to the relatively short data length causes the effective data rate to be rather low.

4.5.2 Frame synchronization

4.5.2.1 Character-oriented

Character-oriented framing assumes that character synchronization has already been achieved by the hardware. The sender uses special characters to indicate the start and end of frames, and may also use them to indicate header boundaries and to assist the receiver gain character synchronization. Frames must be of an integral character length.

Most commonly, a DLE (datalink escape) character is used to signal that the next character is a control character, with DLE SOH (start of header) used to indicate the start of the frame (it starts with a header), DLE STX (start of text) used to indicate the end of the header and start of the data portion, and DLE ETX (end of text) used to indicate the end of the frame.

When a DLE character occurs in the header or the data portion of a frame, the sender must somehow let the receiver know that it is not intended to signal a control character. The sender does this by inserting an extra DLE character after the one occurring inside the frame, so that when the receiver sees two DLEs in a row, it knows to delete one and interpret the other as header or data.

4.5.2.2 Bit-oriented

Bit-oriented framing only assumes that bit synchronization has been achieved by the underlying hardware, and the incoming bit stream is scanned at all possible bit positions for special patterns generated by the sender. The sender uses a special pattern (a flag pattern) to delimit frames (one flag at each end). A commonly used flag pattern is HDLC's 01111110 flag.

If the flag pattern appears anywhere in the header or data of a frame, then the receiver may prematurely detect the start or end of the received frame. To combat this, the sender makes sure that the frame body it sends has no flags in it at any position (note that since there is no character synchronization, the flag pattern can start at any bit location within the stream). It does this by bit-stuffing, inserting an extra bit in any pattern that is beginning to look like a flag. In HDLC, whenever 5 consecutive 1'a are encountered in the data, a 0 is inserted after the 5th 1, regardless of the next bit in the data. On the receiving end, the bit stream is piped through a shift register as the receiver looks for the flag pattern. If 5 consecutive 1's followed by a 0 is seen, then the 0 is dropped before sending the data on (the receiver destuffs the stream). If 6 1's and a 0 is seen, it is a flag and either the current frame is ended or a new frame is started, depending on the current state of the receiver. If more than 6 consecutive 1's are seen, then the receiver has detected an invalid pattern, and usually the current frame, if any, is discarded.

4.6 Error Correction

In addition to receiving the data in logical units called frames, the receiver should have some way of determining if the data has been corrupted or not. If it has been corrupted, it is desirable not only to realize that, but to make an attempt to obtain the correct data. This process is called error correction.

The aim of an error detection technique is to enable the receiver of a message transmitted through a noisy (error-introducing) channel to determine whether the message has been corrupted. To do this, the transmitter constructs a value (called a checksum) that is a function of the message, and appends it to the message. The receiver can then use the same function to calculate the checksum of the received message and compare it with the appended checksum to see if the message was correctly received. For example, if we chose a checksum function which was simply the sum of the bytes in the message mod 256 (i.e. modulo 256), then it might go something as follows.

Message : 6 23 4

Message with checksum : 6 23 4 33

Message after transmission : 6 27 4 33

In the above, the second byte of the message was corrupted from 23 to 27 by the communications channel. However, the receiver can detect this by comparing the transmitted checksum (33) with the computer checksum of 37 (6 + 27 + 4). If the checksum itself is corrupted, a correctly transmitted message might be incorrectly identified as a corrupted one. However, this is a safe-side failure. A dangerous-side failure occurs where the message and/or checksum is corrupted in a manner that results in a transmission that is internally consistent. Unfortunately, this possibility is completely unavoidable and the best that can be done is to minimize its probability by increasing the amount of information in the checksum (e.g. widening the checksum from one byte to two bytes).

4.6.1 Cyclic Redundancy Check

To strengthen the checksum, we could change from an 8-bit register to a 16-bit register (i.e. sum the bytes mod 65536 instead of mod 256) so as to apparently reduce the probability of failure from 1/256 to 1/65536. While basically a good idea, it fails in this case because the formula used is not sufficiently "random"; with a simple summing formula, each incoming byte affects roughly only one byte of the summing register no matter how wide it is. For example, in the second example above, the summing register could be a Megabyte wide, and the error would still go undetected. This problem can only be solved by replacing the simple summing formula with a more sophisticated formula that causes each incoming byte to have an effect on the entire checksum register.

Thus, we see that at least two aspects are required to form a strong checksum function:

WIDTH: A register width wide enough to provide a low a-priori probability of failure (e.g. 32-bits gives a 1/2^32 chance of failure).

CHAOS: A formula that gives each input byte the potential to change any number of bits in the register.

A powerful method for detecting errors in the received data is by grouping the bytes of data into a block and calculating a Cyclic Redundancy Check (CRC). This is usually done by the data link protocol and calculated CRC is appended to the end of the data link layer frame. Protocols at the network layer and higher (e.g. IP, UDP, TCP) usually use a simpler checksum to verify that the data being transported has not been corrupted by the processing performed by the nodes in the network.

The basic idea of CRC algorithms is simply to treat the message as an enormous binary number, to divide it by another fixed binary number, and to make the remainder from this division the checksum. Upon receipt of the message, the receiver can perform the same division and compare the remainder with the "checksum" (transmitted remainder).

For a n-bit CRC, the divisor is actually a (n+1)-bit number. The remainder (which is used for the CRC) is the n-bit value. The divisor is usually described mathematically as a generator polynomial ($x^{16}+x^{15}+x^2+x^0$ for CRC-16), which is why they are not usually represented as bit strings.

4.6.1.1 Polynomial Arithmetic

While the division scheme described in the previous section is very similar to the checksumming schemes called CRC schemes, the CRC schemes are slightly different, and we need to delve into some strange number systems to understand them.

The word you will hear all the time when dealing with CRC algorithms is the word "polynomial". A given CRC algorithm will be said to be using a particular polynomial, and CRC algorithms in general are said to be operating using polynomial arithmetic. What does this mean?

Instead of the divisor, dividend (message), quotient, and remainder (as described in the previous section) being viewed as positive integers, they are viewed as polynomials with binary coefficients. This is done by treating each number as a bit-string whose bits are the coefficients of a polynomial. For example, the ordinary number 23 (decimal) is 17 (hex) and 10111 binary and so it corresponds to the polynomial:

$1*x^4+0*x^3+1*x^2+1*x^1+1*x^0$

or, more simply:

$x^4+x^2+x^1+x^0$

Using this technique, the message, and the divisor can be represented as polynomials and we can do all our arithmetic just as before, except that now it's all cluttered up with Xs. For example, suppose we wanted to multiply 1101 by 1011. We can do this simply by multiplying the polynomials:

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

At this point, to get the right answer, we have to pretend that x is 2 and propagate binary carries from the $3x^3$ yielding

$x^7+x^3+x^2+x^1+x^0$

It's just like ordinary arithmetic except that the base is abstracted and brought into all the calculations explicitly instead of being there implicitly. So what's the point?

The point is that *if* we pretend that we *don't* know what x is, we *cannot* perform the carries. We don't know that $3x^3$ is the same as x^4+x^3 because we don't know that x is 2. In this true polynomial arithmetic the relationship between all the coefficients is unknown and so the coefficients of each power effectively become strongly typed; coefficients of x^2 are effectively of a different type to coefficients of x^3.

With the coefficients of each power nicely isolated, mathematicians came up with all sorts of different kinds of polynomial arithmetics simply by changing the rules about how coefficients work. Of these schemes, one in particular is relevant here, and that is a polynomial arithmetic where the coefficients are calculated MOD 2 and there is no carry; all coefficients must be either 0 or 1 and no carries are calculated. This is called "polynomial arithmetic mod 2". Under the other arithmetic, the $3x^3$ term was propagated using the carry mechanism using the knowledge that x=2. Under "polynomial arithmetic mod 2", we don't know what x is, there are no carries, and all coefficients have to be calculated mod 2. Thus, the result becomes:

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

Polynomial arithmetic mod 2 is just binary arithmetic mod 2 with no carries. The arithmetic performed during CRC calculations is performed in binary with no carries (CRC arithmetic).

In fact, both addition and subtraction in CRC arithmetic is equivalent to the XOR operation, and the XOR operation is its own inverse. This effectively reduces the operations of the first level of power (addition, subtraction) to a single operation that is its own inverse. This is a very convenient property of the arithmetic.

Multiplication is absolutely straightforward, being the sum of the first number, shifted in accordance with the second number.

```
   1101
 x 1011
   ----
   1101
  1101.
 0000..
1101...
-------
1111111   Note: The sum uses CRC addition
-------
```

Division is a little messier as we need to know when "a number goes into another number". To do this, we invoke the weak definition of magnitude defined earlier: that X is greater than or equal to Y if and only if the position of the highest 1 bit of X is the same or greater than the position of the highest 1 bit of Y. Here's a fully worked division.

```
                1100001010
              ____________
10011   ) 11010110110000
          10011,,,,....
          _____,,,....
           10011,,,....
           10011,,,....
           _____,,,....
            00001,,,....
            00000,,,....
            _____,,,....
             00010,,....
             00000,,....
             _____,,....
              00101,....
              00000,....
              _____,....
               01011....
               00000....
               _____....
                10110...
                10011...
                _____...
                 01010..
                 00000..
                 _____..
                  10100.
                  10011.
                  _____.
                   01110
                   00000
                   _____
                    1110   = Remainder
```

To perform a CRC calculation, we need to choose a divisor. In mathematics speak the divisor is called the "generator polynomial" or simply the "polynomial", and is a key parameter of any CRC algorithm. It would probably be more friendly to call the divisor something else, but the polynomial talk is so deeply ingrained in the field that it would now be confusing to avoid it. As a compromise, we will refer to the CRC polynomial as the "poly".

You can choose any poly and come up with a CRC algorithm. However, some polys are better than others, and so it is wise to stick with the tried an tested ones.

The width (position of the highest 1 bit) of the poly is very important as it dominates the whole calculation. Typically, widths of 16 or 32 are chosen so as to simplify implementation on modern computers. The width of a poly is the actual bit position of the highest bit. For example, the width of 10011 is 4, not 5.

Having chosen a poly, we can proceed with the calculation. This is simply a division (in CRC arithmetic) of the message by the poly. The only trick is that W zero bits are appended to the message before the CRC is calculated.

The division yields a quotient, which we throw away, and a remainder, which is the calculated checksum. This ends the calculation.

Usually, the checksum is then appended to the message and the result transmitted.

At the other end, the receiver can do one of two things:

1. Separate the message and checksum. Calculate the checksum for the message (after appending W zeros) and compare the two checksums.

2. Checksum the whole lot (without appending zeros) and see if it comes out as zero!

4.6.1.2 Choosing A Poly

Some popular polys are:

- (16,12,5,0) [X25 standard]
- (16,15,2,0) ["CRC-16"]
- (32,26,23,22,16,12,11,10,8,7,5,4,2,1,0) [Ethernet]

The CRC-16 is able to detect all single errors, all double errors, all odd numbers of errors and all errors with burst less than 16 bits in length. In addition 99.9984 % of other error patterns will be detected.

The transmitted message T is a multiple of the poly. To see this, note that

1. the last W bits of T is the remainder after dividing the augmented (by zeros remember) message by the poly, and
2. addition is the same as subtraction so adding the remainder pushes the value up to the next multiple.

Now note that if the transmitted message is corrupted in transmission that we will receive T+E where E is an error vector (and + is CRC addition (i.e. XOR)). Upon receipt of this message, the receiver divides T+E by G. As T mod G is 0, (T+E) mod G = E mod G. Thus, the capacity of the poly we choose to catch particular kinds of errors will be determined by the set of multiples of G, for any corruption E that is a multiple of G will be undetected. Our task then is to find classes of G whose multiples look as little like the kind of line noise (that will be creating the corruptions) as possible. So let's examine the kinds of line noise we can expect.

4.6.1.2.1 Single Bit Errors A single bit error means E=1000...0000. We can ensure that this class of error is always detected by making sure that G has at least two bits set to 1. Any multiple of G will be constructed using shifting and adding and it is impossible to construct a value with a single bit by shifting an adding a single value with more than one bit set, as the two end bits will always persist.

4.6.1.2.2 Two Bit Errors To detect all errors of the form 100...000100...000 (i.e. E contains two 1 bits) choose a G that does not have multiples that are 11, 101, 1001, 10001, 100001, etc.

4.6.1.2.3 Errors with an odd number of bits We can catch all corruptions where E has an odd number of bits by choosing a G that has an even number of bits. To see this, note that

1. CRC multiplication is simply XORing a constant value into a register at various offsets,
2. XORing is simply a bit-flip operation, and
3. if you XOR a value with an even number of bits into a register, the oddness of the number of 1 bits in the register remains invariant.

4.6.1.2.4 Burst Errors A burst error looks like E=000...000111...11110000...00. That is, E consists of all zeros except for a run of 1s somewhere inside. This can be recast as E=(10000...00)(1111111...111) where there are z zeros in the LEFT part and n ones in the RIGHT part. To catch errors of this kind, we simply set the lowest bit of G to 1. Doing this ensures that LEFT cannot be a factor of G. Then, so long as G is wider than RIGHT, the error will be detected.

4.7 Transmission over Fiber

4.7.1 Structure of the cable

Basically, a fiber optic cable is composed of two concentric layers termed the core and the cladding. The core and cladding have different indices of refraction with the core having n_1 and the cladding n_2. Light is piped through the core. A fiber optic cable has an additional coating around the cladding called the jacket. The jacket usually consists of one or more layers of polymer. Its role is to protect the core and cladding from shocks that might affect their optical or physical properties. It acts as a shock absorber. The jacket also provides protection from abrasions, solvents and other contaminants. The jacket does not have any optical properties that might affect the propagation of light within the fiber optic cable. There may be a strength member added to the fiber optic cable so that it can be pulled during installation.

How is light guided down the fiber optic cable in the core? This occurs because the core and cladding have different indices of refraction with the index of the core, n_1, always being greater than the index of the cladding, n_2. If a light ray is injected and strikes the core-to-cladding interface at an angle greater than the critical angle then it is reflected back into the core. Since the angle of incidence is always equal to the angle of reflection the reflected light will again be reflected. The light ray will then continue this bouncing path down the length of the fiber optic cable. If the light ray strikes the core-to-cladding interface at an angle less than the critical angle then it passes into the cladding where it is attenuated very rapidly with propagation distance.

Light can be guided down the fiber optic cable if it enters at less than the critical angle. This angle is fixed by the indices of refraction of the core and cladding and is given by the formula:

$Q_c = \arccos(\frac{n_2}{n_1})$

When it comes to size, fiber optic cables have exceedingly small diameters. To get some feeling for how small these sizes actually are, understand that a human hair has a diameter of 100 microns. Fiber optic cable sizes are usually expressed by first giving the core size followed by the cladding size. Consequently, 50/125 indicates a core diameter of 50 microns and a cladding diameter of 125 microns; 100/140 indicates a core diameter of 100 microns and a cladding diameter of 140 microns. The larger the core the more light can be coupled into it from external acceptance angle cone. However, larger diameter cores may actually allow too much light in and too much light may cause Receiver saturation problems. The 8/125 cable is often found when a fiber optic data link operates with single-mode propagation. The 62.5/125 cable, is often found in a fiber optic data link that operates with multi-mode propagation.

Attenuation is principally caused by two physical effects, absorption and scattering. Absorption removes signal energy in the interaction between the propagating light (photons) and molecules in the core. Scattering redirects light out of the core to the cladding.

Glass fiber optic cable has the lowest attenuation and comes at the highest cost. A pure glass fiber optic cable has a glass core and a glass cladding. During the glass fiber optic cable fabrication process impurities are purposely added to the pure glass so as to obtain the desired indices of refraction needed to guide light. Germanium or phosphorous are added to increase the index of refraction. Boron or fluorine is added to decrease the index of refraction.

Plastic fiber optic cable has the highest attenuation, but comes at the lowest cost. Plastic fiber optic cable has a plastic core and plastic cladding.

Plastic Clad Silica (PCS) fiber optic cable has an attenuation that lies between glass and plastic and a cost that lies between their cost as well.

A pulse transmitted over fiber will be attenuated (decrease in amplitude) relative to the input pulse. It also suffers time dispersion (change in width and sharpness). The reasons for this are as follows. The higher order modes, the bouncing rays, tend to leak into the cladding as they propagate down the fiber optic cable. They lose some of their energy into heat. This results in an attenuated output signal. The input pulse is split among the different rays that travel down the fiber optic cable. The bouncing rays and the lowest order mode, traveling down the center axis, are all traversing paths of different lengths from input to output. Consequently, they do not all reach the right end of the fiber optic cable at the same time. When the output pulse is constructed from these separate ray components the result is time dispersion.

When it comes to mode of propagation fiber optic cable can be one of two types, multi-mode or single-mode. These provide different performance with respect to both attenuation and time dispersion.

The single-mode fiber optic cable provides the better performance at, of course, a higher cost. In single-mode fiber the diameter of the core is fairly small relative to the cladding. Typically, the cladding is ten times thicker than the core. Because of this when light enters the fiber optic cable on the right it propagates down toward the left in just a single ray, a single-mode, and the lowest order mode. The higher order modes are absent. Consequently, there is no energy lost to heat by having these modes leak into the cladding. There is little time dispersion, only that due to propagation through the non-zero diameter, single mode cylinder. Single mode propagation exists only above a certain specific wavelength called the cutoff wavelength.

4.7.2 Transmission of data

The technique used to bring about the sharing of a fiber optic cable among a multiplicity of transmission requirements is called multiplexing. Multiplexing is particularly attractive when the transmission medium is fiber optic cable, because the tremendous bandwidth presented by fiber optic cable presents the greatest opportunity for sharing.

There are two techniques for carrying out multiplexing on fiber optic cable in the premise environment. These two techniques are Time Division Multiplexing (TDM) and Wavelength Division Multiplexing (WDM).

With TDM a multiplicity of communication links share the same fiber optic cable on the basis of time. The multiplexer(s) set up a continuous sequence of time slots using clocks. The duration of the time slots depends upon a number of different engineering design factors; most notably the needed transmission speeds for the different links. Each communication link is assigned a specific time slot, a TDM channel, during which it is allowed to send its data from the Source end to the User end. During this time slot no other link is permitted to send data. The multiplexer at the Source end takes in data from the Sources connected to it. It then loads the data from each Source into its corresponding TDM channel. The multiplexer at the User end unloads the data from each channel and sends it to the corresponding User.

With WDM a multiplicity of communication links share the same fiber optic cable on the basis of wavelength. The data stream from each source is assigned an optical wavelength. The multiplexer has within it the modulation and transmission processing circuitry. The multiplexer modulates each data stream from each source. After the modulation process the resulting optical signal generated for each source data stream is placed on its assigned wavelength. The multiplexer then couples the totality of optical signals generated for all source data streams into the fiber optic cable. These different wavelength optical signals propagate simultaneously. This is in contrast to TDM.

At the other end the multiplexer receives these simultaneous optical signals. It separates these signals out according to their different wavelengths by using prisms. This constitutes the demultiplexing operation.

Chapter 5

Network Transmission Standards: Data Link Layer

5.1 Chapter Structure

5.1.1 Outcomes

1. Demonstrate knowledge and understanding of the Data Link protocols covered in this chapter.
2. Be able to compare the strategies employed by each of these Data Link network technologies.
3. Be able to choose an appropriate technology for a given scenario, based on the constraints associated with each Data Link technology and its variations.
4. Assess performance implications of the constraints associated with each Data Link technology and its variations.
5. Appreciate the issues involved (including social) in using and standardizing on any particular Data Link technology.
6. Be able to represent the use of the various Data Link technologies using appropriately annotated diagrams.
7. Be able to classify the networking technologies presented in this chapter according to the criteria discussed in previous chapters.
8. Appreciate the rapid and recent advances in many of these Data Link technologies.

5.1.2 Objectives

This chapter:

1. Describes a range of standardized networking technologies used at the Data Link layer.
2. Provides details of each networking technology, focusing on a common set of attributes.
3. Describes the physical and logical limitations of each Data Link technology.
4. Describes the reason for these limitations in terms of the physical infrastructure and nature of the protocols used.
5. Specifies the standardization process for many of the Data Link technologies, and the issues arising.

6. Shows various configurations of equipment that can be used to implement networks based on these Data Link technologies.

7. Shows trends and recent changes in the most popular of the Data Link technologies.

5.2 Ethernet

Ethernet			
Size	Topology	Cable	Transmission
LAN	Physical Bus or Logical Bus, Physical Star	Coaxial UTP STP Fiber	Packet Switching, using CSMA/CD

Ethernet is a local area network (LAN) technology that transmits information between computers at speeds of 10 and 100 million bits per second (Mbps). Currently the most widely used version of Ethernet technology is the 10-Mbps twisted-pair variety.

The 10-Mbps Ethernet media varieties include the original thick coaxial system, as well as thin coaxial, twisted-pair, and fiber optic systems. The most recent Ethernet standard defines the new 100-Mbps Fast Ethernet system which operates over twisted-pair and fiber optic media.

Each Ethernet-equipped computer, also known as a station, operates independently of all other stations on the network: there is no central controller. All stations attached to an Ethernet are connected to a shared signalling system, also called the medium. Ethernet signals are transmitted serially, one bit at a time, over the shared signal channel to every attached station. To send data a station first listens to the channel, and when the channel is idle the station transmits its data in the form of an Ethernet frame, or packet.

After each frame transmission, all stations on the network must contend equally for the next frame transmission opportunity. This ensures that access to the network channel is fair, and that no single station can lock out the other stations. Access to the shared channel is determined by the medium access control (MAC) mechanism embedded in the Ethernet interface located in each station. The medium access control mechanism is based on a system called Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

The "Multiple Access" part means that every station is connected to a single copper wire (or a set of wires that are connected together to form a single data path). The "Carrier Sense" part says that before transmitting data, a station checks the wire to see if any other station is already sending something. If the LAN appears to be idle, then the station can begin to send data.

The Ethernet system consists of three basic elements:

1. the physical medium used to carry Ethernet signals between computers,

2. a set of medium access control rules embedded in each Ethernet interface that allow multiple computers to fairly arbitrate access to the shared Ethernet channel, and

3. an Ethernet frame that consists of a standardized set of bits used to carry data over the system.

5.2.1 Physical Limitation of Ethernet

First of all, there are distance limitations:

- 10Base2 limited to 185 meters per unrepeated cable segment.
- 10Base5 limited to 500 meters per unrepeated cable segment.
- 10Base-F depends on the signalling technology and medium used but can go up to 2KM.

- 10Base-T generally accepted to have a maximum run of 100-150M, but is really based on signal loss in decibels (11.5DB maximum loss source to destination).
- 10Broad36 limited to 3,600 meters (almost 2.25 miles).

Then there are limitations on the number of repeaters and cable segments allowed between any two stations on the network. There are two different ways of looking at the same rules:

1. The Ethernet way: A remote repeater pair (with an intermediate point-to-point link) is counted as a single repeater (IEEE calls it two repeaters). You cannot put any stations on the point to point link (by definition!), and there can be two repeaters in the path between any pair of stations. This seems simpler than the IEEE terminology, and is equivalent.
2. The IEEE way: There may be no more than five (5) repeated segments, nor more than four (4) repeaters between any two Ethernet stations; and of the five cable segments, only three (3) may be populated. This is referred to as the "5-4-3" rule (5 segments, 4 repeaters, 3 populated segments).

It can really get messy when you start cascading through 10Base-T hubs, which are repeaters unto themselves. Just try to remember, that any possible path between two network devices on an unbridged/unrouted network cannot pass through more than 4 repeaters or hubs, nor more than 3 populated cable segments.

Finally, 10Base2 is limited to a maximum of 30 network devices per unrepeated network segment with a minimum distance of 0.5m between T-connectors. 10Base5 is limited to a maximum of 100 network devices per unrepeated segment, with a minimum distance of 2.5m between taps/T's (usually indicated by a marker stamped on the cable itself every 2.5m). 10Base-T and 10Base-F are star-wired, so there is no minimum distance requirement between devices, since devices cannot be connected serially. You can install up to the Ethernet maximum of 1024 stations per network with both 10Base-T and 10Base-F.

10Base2 (thin Ethernet or Cheapernet) is the least expensive way to cable an Ethernet network. However, the price difference between 10Base2 and 10Base-T (Ethernet over UTP) is rapidly diminishing. Still, for small, budget-conscious installations, 10Base2 is the most economical topology. The disadvantages of 10Base2 is that any break in the cable or poor connection will bring the entire network down, and you need repeaters if you have more than 30 devices connected to the network or the cable length exceeds 185 meters (607 feet).

10Base-T is the most flexible topology for LANs, and is generally the best choice for most network installations. 10Base-T hubs, or multi-hub concentrators, are typically installed in a central location to the user community, and inexpensive UTP cabling is run to each network device (which may be 100m from the hub). The signalling technology is very reliable, even in somewhat noisy environments, and 10Base-T hubs will usually detect many network error conditions and automatically shut-down the offending port(s) without affecting the rest of the network (unless, of course, the offending port was your server, shared printer, or router to the rest of the world). While the hardware is more expensive than 10Base2, the cabling is cheaper and requires less skill to install, making 10Base-T installation costs only slightly higher than 10Base2. The flexibility and reliability more than offset the marginally higher price.

5.2.2 Access and Collisions

An Ethernet station sends data at a rate of 10 megabits per second. That bit allows 100 nanoseconds per bit. Light and electricity travel about one foot in a nanosecond. Therefore, after the electric signal for the first bit has traveled about 100 feet down the wire, the station has begun to send the second bit. However, an Ethernet cable can run for hundreds of feet. If two stations are located, say, 250 feet apart on the same cable, and both begin transmitting at the same time, then they will be in the middle of the third bit before the signal from each reaches the other station.

This explains the need for the "Collision Detect" part. Two stations can begin to send data at the same time, and their signals will "collide" nanoseconds later. When such a collision occurs, the two stations stop transmitting, "back off", and try again later after a randomly chosen delay period.

It's unfortunate that the original Ethernet design used the word "collision" for this aspect of the Ethernet medium access control mechanism. If it had been called something else, such as "stochastic arbitration

event (SAE)," then no one would worry about the occurrence of SAEs on an Ethernet. However, "collision" sounds like something bad has happened, leading many people to think that collisions are an indication of network failure.

The truth of the matter is that collisions are absolutely normal and expected events on an Ethernet, and simply indicate that the CSMA/CD protocol is functioning as designed. As more computers are added to a given Ethernet, and as the traffic level increases, more collisions will occur as part of the normal operation of an Ethernet.

The design of the system ensures that the majority of collisions on an Ethernet that is not overloaded will be resolved in microseconds, or millionths of a second. A normal collision does not result in lost data. In the event of a collision the Ethernet interface backs off (waits) for some number of microseconds, and then automatically retransmits the data.

On a network with heavy traffic loads it may happen that there are multiple collisions for a given frame transmission attempt. This is also normal behavior. If repeated collisions occur for a given transmission attempt, then the stations involved begin expanding the set of potential back-off times from which they chose their random retransmission time.

Repeated collisions for a given packet transmission attempt indicate a busy network. The expanding back-off process, formally known as "truncated binary exponential back-off," is a clever feature of the Ethernet MAC that provides an automatic method for stations to adjust to traffic conditions on the network. Only after 16 consecutive collisions for a given transmission attempt will the interface finally discard the Ethernet packet. This can happen only if the Ethernet channel is overloaded for a fairly long period of time, or is broken in some way.

While an Ethernet can be built using one common signal wire, such an arrangement is not flexible enough to wire most buildings. Unlike an ordinary telephone circuit, Ethernet wire cannot be just spliced together, connecting one copper wire to another. Ethernet requires a repeater. A repeater is a simple station that is connected to two wires. Any data that it receives on one wire it repeats bit-for-bit on the other wire. When collisions occur, it repeats the collision as well.

In common practice, repeaters are used to convert the Ethernet signal from one type of wire to another. In particular, when the connection to the desktop uses ordinary telephone wire, the hub back in the telephone closet contains a repeater for every phone circuit. Any data coming down any phone line is copied onto the main Ethernet coax cable, and any data from the main cable is duplicated and transmitted down every phone line. The repeaters in the hub electrically isolate each phone circuit, which is necessary if a 10 megabit signal is going to be carried 300 feet on ordinary wire.

Every set of rules is best understood by characterizing its worst case. The worst case for Ethernet starts when a PC at the extreme end of one wire begins sending data. The electric signal passes down the wire through repeaters, and just before it gets to the last station at the other end of the LAN, that station (hearing nothing and thinking that the LAN is idle) begins to transmit its own data. A collision occurs. The second station recognizes this immediately, but the first station will not detect it until the collision signal retraces the first path all the way back through the LAN to its starting point.

Any system based on collision detect must control the time required for the worst round trip through the LAN. As the term "Ethernet" is commonly defined, this round trip is limited to 50 microseconds (millionths of a second). At a signalling speed of 10 million bits per second, this is enough time to transmit 500 bits. At 8 bits per byte, this is slightly less than 64 bytes.

To make sure that the collision is recognized, Ethernet requires that a station must continue transmitting until the 50 microsecond period has ended. If the station has less than 64 bytes of data to send, then it must pad the data by adding zeros at the end.

In simpler days, when Ethernet was dominated by heavy duty coax cable, it was possible to translate the 50 millisecond limit and other electrical restrictions into rules about cable length, number of stations, and number of repeaters. However, by adding new media (such as Fiber Optic cable) and smarter electronics, it becomes difficult to state physical distance limits with precision. However those limits work out, they are ultimately reflections of the constraint on the worst case round trip.

It would be possible to define some other Ethernet-like collision system with a 40 microsecond or 60 microsecond period. Changing the period, the speed, and the minimum message size simply require a new standard and some alternate equipment. AT&T, for example, once promoted a system called "Starlan" that transmitted data at 1 megabit per second over older phone wire. Many such systems are possible, but the

term "Ethernet" is generally reserved for a system that transmits 10 megabits per second with a round trip delay of 50 microseconds.

To extend the LAN farther than the 50 microsecond limit will permit, one needs a bridge or router. These terms are often confused:

A repeater receives and then immediately retransmits each bit. It has no memory and does not depend on any particular protocol. It duplicates everything, including the collisions. A bridge receives the entire message into memory. If the message was damaged by a collision or noise, then it is discarded. If the bridge knows that the message was being sent between two stations on the same cable, then it discards it. Otherwise, the message is queued up and will be retransmitted on another Ethernet cable. The bridge has no address. Its actions are transparent to the client and server workstations. A router acts as an agent to receive and forward messages. The router has an address and is known to the client or server machines. Typically, machines directly send messages to each other when they are on the same cable, and they send the router messages addressed to another zone, department, or subnetwork. Routing is a function specific to each protocol. For IPX, the Novell server can act as a router. For SNA, an APPN Network Node does the routing. TCP/IP can be routed by dedicated devices, UNIX workstations, or OS/2 servers.

5.2.3 Ethernet Control Procedure

The Ethernet control procedure defines how and when a station may transmit packets into the common cable (see Figure 5.1). The key purpose is fair resolution of occasional contention among transmission stations.

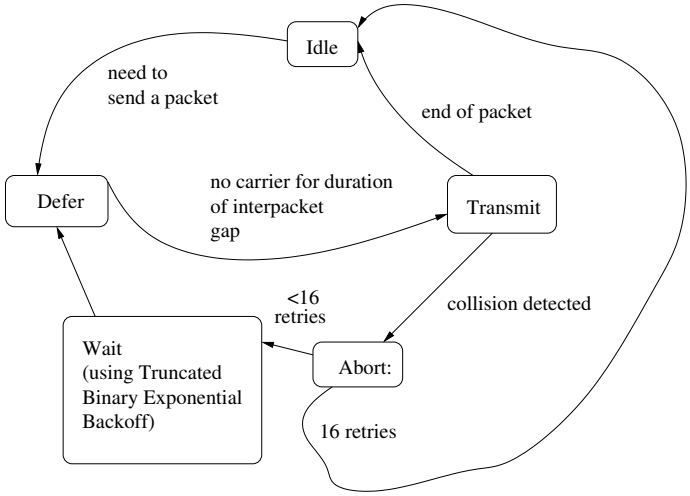


Figure 5.1: Ethernet Control Procedure

Defer: A station must not transmit into the coaxial cable when carrier is present or within the minimum packet spacing time after carrier has ended.

Transmit: A station may transmit if it is not deferring. It may continue to transmit until either the end of the packet is reached or a collision is detected.

Abort: If a collision is detected, transmission of the packet must terminate, and a jam (4-6 bytes of arbitrary data) is transmitted to ensure that all other participants in the collision also recognize its occurrence.

Retransmit: After a station has detected a collision and aborted. It must wait for a random retransmission delay, defer as usual, and then attempt to retransmit the packet. The random time interval is computed using the back-off algorithm (below). After 16 transmission attempts, a higher level (e.g. software) decision is made to determine whether to continue or abandon the effort.

Retransmission delays are computed using the Truncated Binary Exponential Back-off algorithm, with the aim of fairly resolving contention among up to 1024 stations. The delay (the number of time units) before the nth attempt is a uniformly distributed random number from [0, 2n-1] for 0 < n <= 10 (n = 0 is the original attempt). For attempts 11 - 15, the interval is truncated and remains at [0, 1023]. The unit of time for the retransmission delay is 512 bit time (51.2 microseconds).

When the network is unloaded and collisions are rare, the mean seldom departs from one and retransmissions are prompt. As the traffic load increases, more collisions are experienced, a backlog of packets builds up in the stations, retransmission intervals increase, and retransmission traffic backs off to sustain channel efficiency.

5.2.4 The Ethernet Frame

The heart of the Ethernet system is the Ethernet frame, which is used to deliver data between computers. The frame consists of a set of bits organized into several fields. These fields include address fields, a variable size data field that carries from 46 to 1,500 bytes of data, and an error checking field that checks the integrity of the bits in the frame to make sure that the frame has arrived intact.

The Ethernet frame encapsulates payload data by adding a 14 byte header before the data and appending a 4-byte (32-bit) Cyclic Redundancy Check (CRC) after the data. The entire frame is preceded by a small idle period (the minimum inter-frame gap, 9.6 microsecond (ţS)) and a 8 byte preamble.

62 bits	Preamble +
2 bits	Start of frame
6 bytes	Destination Addr
6 bytes	Source Addr
2 bytes	Length or Type
46-1500 bytes	Data
4 bytes	Frame Check Sequence

Table 5.1: Structure of an Ethernet packet

The structure of an Ethernet packet is shown in Table 5.1. The Ethernet packet preamble is normally generated by the chipset. Software is responsible for the destination address, source address, type, and data. The chips normally will append the frame check sequence.

5.2.4.1 Preamble

The purpose of the idle time before transmission starts is to allow a small time interval for the receiver electronics in each of the nodes to settle after completion of the previous frame. A node starts transmission by sending an 8 byte (64 bit) preamble sequence. This consists of 62 alternating 1's and 0's followed by the pattern 11. When encoded using Manchester encoding, the 62 alternating bits produce a 10 MHz square wave.

The purpose of the preamble is to allow time for the receiver in each node to achieve lock of the receiver Digital Phase Lock Loop which is used to synchronize the receive data clock to the transmit data clock. At the point when the first bit of the preamble is received, each receiver may be in an arbitrary state (i.e. have an arbitrary phase for its local clock). During the course of the preamble it learns the correct phase, but in so doing it may miss (or gain) a number of bits. A special pattern (11), known as the start of frame delimiter, is therefore used to mark the last two bits of the preamble. When this is received, the Ethernet receive interface starts collecting the bits into bytes for processing by the MAC layer.

5.2.4.2 Header

The Destination Ethernet Address is the address of the intended receiver. The broadcast address is all 1 bits. The Source Ethernet Address is the unique Ethernet address of the sending station. The Length or

Type field, for IEEE 802.3 is the number of bytes of data. For Ethernet I&II this is the type of packet. Types codes are > 1500 to allow both to coexist. The type code for IP packets is 0x800. Short packets must be padded to 46 bytes.

5.2.4.3 CRC

The 32-bit CRC added at the end of the frame provides error detection in the case where line errors (or transmission collisions in Ethernet) result in corruption of the frame. Any frame with an invalid CRC is discarded by the receiver without further processing. The protocol does not provide any indication that a frame has been discarded due to an invalid CRC.

The Frame Check Sequence is a 32 bit CRC calculated using the AUTODIN II polynomial. This field is normally generated by the chip.

5.2.5 Ethernet Frame Formats

The minimum frame payload is 46 Bytes (dictated by the slot time of the Ethernet LAN architecture). The maximum frame rate is achieved by a single transmitting node which does not therefore suffer any collisions. This implies a frame consisting of 72 Bytes (see table above) with a 9.6 ţs inter-frame gap (corresponding to 12 Bytes at 10 Mbps). The total utilized period (measured in bits) corresponds to 84 Bytes.

Frame Part	Minimum Size Frame	Maximum Size Frame
Inter Frame Gap (9.6ţs)	12 Bytes	12 Bytes
MAC Preamble (+ SFD)	8 Bytes	8 Bytes
MAC Destination Address	6 Bytes	6 Bytes
MAC Source Address	6 Bytes	6 Bytes
MAC Type (or Length)	2 Bytes	2 Bytes
Payload (Network PDU)	46 Bytes	1500 Bytes
Check Sequence (CRC)	4 Bytes	4 Bytes
Total Frame Physical Size	84 Bytes	1538 Bytes

The two address fields in the frame carry 48-bit addresses, called the destination and source addresses. The IEEE controls the assignment of these addresses by administering a portion of the address field. The IEEE does this by providing 24-bit identifiers called "Organizationally Unique Identifiers" (OUIs), since a unique 24-bit identifier is assigned to each organization that wishes to build Ethernet interfaces. The organization, in turn, creates 48-bit addresses using the assigned OUI as the first 24 bits of the address. This 48-bit address is also known as the physical address, hardware address, or MAC address.

The source address field of each frame must contain the unique address (universal or local) assigned to the sending card. The destination field can contain a "multicast" address representing a group of workstations with some common characteristic. A Novell client may broadcast a request to identify all Netware servers on the LAN, while a Microsoft or IBM client machine broadcasts a query to all machines supporting NETBIOS to find a particular server or domain.

In normal operation, an Ethernet adapter will receive only frames with a destination address that matches its unique address, or destination addresses that represent a multicast message. However, most Ethernet adapters can be set into "promiscuous" mode where they receive all frames that appear on the LAN. If this poses a security problem, a new generation of smart hub devices can filter out all frames with private destination addresses belonging to another station.

There are four common conventions for the format of the remainder of the frame:

1. Ethernet II or DIX
2. IEEE 802.3 and 802.2
3. 802.3 SNAP

4. Raw 802.3

5.2.5.1 Ethernet II or DIX

Before the development of international standards, Xerox administered the Ethernet conventions. As each vendor developed a protocol, a two byte Type code was assigned by Xerox to identify it. Codes were given out to XNS (the Xerox own protocol), DECNET, IP, and Novell IPX. Since short Ethernet frames must be padded with zeros to a length of 64 bytes, each of these higher level protocols required either a larger minimum message size or an internal length field that can be used to distinguish data from padding. Type field values of particular note include:

Destination Addr	Source Addr	Type

Table 5.2: Ethernet II (DIX) header

0x0600 XNS (Xerox)

0x0800 IP (the Internet protocol)

0x6003 DECNET

5.2.5.2 IEEE 802.3 and 802.2

The IEEE 802 committee was charged to develop protocols that could operate the same way across all LAN media. To allow collision detect, the 10 megabit Ethernet requires a minimum packet size of 64 bytes. Any shorter message must be padded with zeros. The requirement to pad messages is unique to Ethernet and does not apply to any other LAN media. In order for Ethernet to be interchangeable with other types of LANs, it would have to provide a length field to distinguish significant data from padding.

The DIX standard did not need a length field because the vendor protocols that used it (XNS, DECNET, IPX, IP) all had their own length fields. However, the 802 committee needed a standard that did not depend on the good behavior of other programs. The 802.3 standard therefore replaced the two byte type field with a two byte length field.

Xerox had not assigned any important types to have a decimal value below 1500. Since the maximum size of a packet on Ethernet is 1500 bytes, there was no conflict or overlap between DIX and 802 standards. Any Ethernet packet with a type/length field less than 1500 is in 802.3 format (with a length) while any packet in which the field value is greater than 1500 must be in DIX format (with a type).

The 802 committee then created a new field to substitute for Type. The 802.2 header follows the 802.3 header (and also follows the comparable fields in a Token Ring, FDDI, or other types of LAN).

The 802.2 header is three bytes long for control packets or the kind of connectionless data sent by all the old DIX protocols. A four byte header is defined for connection oriented data, which refers primarily to SNA and NETBEUI. The first two bytes identify the SAP. Even with hindsight it is not clear exactly what the IEEE expected this field to be used for. In current use, the two SAP fields are set to 0x0404 for SNA and 0xE0E0 for NETBEUI.

DSAP is the destination service access point. SSAP is the source service access point.

The Service Access Point (SAP) fields provide a demultiplexing capability somewhat analogous to the Ethernet protocol type code. Since these fields are only eight bits, the demultiplexing capability provided is quite limited, so one SAP value (the SNAP SAP, AA hex or 170 decimal) was reserved for an extension to 802.2 LLC called the Sub-Network Access Protocol (SNAP)

Destination Addr	Source Addr	Length

802.3 header

DSAP	SSAP	Control

802.2 header following 802.3 header

Table 5.3: The 802.3 and 802.2 headers

5.2.5.3 SNAP

The IEEE left all the other protocols in a confusing situation. They did not need any new services and did not benefit from the change. Furthermore, a one byte SAP could not substitute for the two byte type field. Yet 802.2 was an International Standard, and that has the force of law in many areas. The compromise was to create a special version of the 802.2 header that conformed to the standard but actually repackaged the old DIX conventions.

Under SNAP, the 802.2 header appears to be a datagram message (control field 0x03) between SAP ID 0xAA. The first five bytes of what 802.2 considers data are actually a sub-header ending in the two byte DIX type value. Any of the old DIX protocols can convert their existing logic to legal 802 SNAP by simply moving the DIX type field back eight bytes from its original location.

AA	AA	03	XX XX XX	DIX-Type

Table 5.4: 802.2 header used under SNAP

5.2.5.4 Raw 802.3

Another frame format that may be present on IEEE 802.3 networks is Novell's raw 802.3 format.

Destination Addr	Source Addr	Length	FFFF (previously checksum)

Table 5.5: Novell's Raw 802.3 Header

This format conforms to the IEEE 802.3 physical layer standard, but does not conform to the IEEE 802.2 datalink layer standard. The historical reason for this is that at the time it was adopted by Novell, IEEE 802.3 had been adopted as a standard but IEEE 802.2 had not. Because of the lack of a datalink header, it is ostensibly suitable for use only on single protocol networks (in this case, IPX only). However, due to the widespread use of Novell's IPX protocol, many manufactures have "bent" the standards to accommodate this frame format, as described in the following paragraph.

Novell's "raw 802.3" framing may be distinguished from IEEE 802.3/802.2 by the presence of FF hex in what would normally be the 802.2 DSAP and SSAP fields in a standard 802.3/802.2 packet. These values are reserved for the DSAP and SSAP by the IEEE, but this particular combination makes no sense when interpreted as a DSAP and SSAP. Note that the IPX checksum field is part of the network layer protocol header (the IPX header), and this method requires that IPX checksumming be disabled (FFFF hex in the IPX checksum field indicates "checksumming disabled" to IPX). With future versions of Netware, IPX checksumming will be available as an option. However, it will be incompatible with this "raw 802.3" frame type. Thus, the "raw 802.3" frame format is well on its way to becoming obsolete, and should be avoided if possible.

5.2.6 State of the Ethernet

The following terms relate to the condition of the Ethernet cable, and the state of communication on it.

5.2.6.1 Collision

SQE is the IEEE term for a collision. (Signal Quality Error). A condition where two devices detect that the network is idle and end up trying to send packets at exactly the same time. (within 1 round-trip delay) Since only one device can transmit at a time, both devices must back off and attempt to retransmit again.

The retransmission algorithm requires each device to wait a random amount of time, so the two are very likely to retry at different times, and thus the second one will sense that the network is busy and wait until the packet is finished. If the two devices retry at the same time (or almost the same time) they will collide again, and the process repeats until either the packet finally makes it onto the network without collisions, or 16 consecutive collision occur and the packet is aborted.

Ethernet is a CSMA/CD (Carrier Sense Multiple Access/ Collision Detect) system. It is possible to not sense carrier from a previous device and attempt to transmit anyway, or to have two devices attempt to transmit at the same time; in either case a collision results. Ethernet is particularly susceptible to performance loss from such problems when people ignore the "rules" for wiring Ethernet. If your network is slowing down and you notice the percentage of collisions is on the high side, you may want try segmenting your network with either a bridge or router to see if performance improves.

5.2.6.2 Late Collision

A late collision occurs when two devices transmit at the same time, but due to cabling errors (most commonly, excessive network segment length or repeaters between devices) neither detects a collision. The reason this happens is because the time to propagate the signal from one end of the network to another is longer than the time to put the entire packet on the network, so the two devices that cause the late collision never see that the other's sending until after it puts the entire packet on the network. Late collisions are detected by the transmitter after the first "slot time" of 64 byte times. They are only detected during transmissions of packets longer than 64 bytes. It's detection is exactly the same as for a normal collision; it just happens "too late."

Typical causes of late collisions are segment cable lengths in excess of the maximum permitted for the cable type, faulty connectors or improper cabling, excessive numbers of repeaters between network devices, and defective Ethernet transceivers or controllers.

Another bad thing about late collisions is that they occur for small packets also, but cannot be detected by the transmitter. A network suffering a measurable rate of late collisions (on large packets) is also suffering lost small packets. The higher protocols do not cope well with such losses. Well, they cope, but at much reduced speed. A 1% packet loss is enough to reduce the speed of NFS by 90% with the default retransmission timers. That's a 10X amplification of the problem.

Finally, Ethernet controllers do not retransmit packets lost to late collisions.

5.2.6.3 The InterFrame Gap

The InterPacket Gap (more properly referred to as the InterFrame Gap, or IFG) is an enforced quiet time of 9.6 us between transmitted Ethernet frames.

5.2.6.4 Promiscuous mode

Promiscuous mode is a condition where the network interface controller will pass all frames, regardless of destination address, up to the higher level network layers. Normally the network controller will only pass up frames that have that device's destination address. However, when put in promiscuous mode, all frames are passed on up the network stack regardless of destination address. Promiscuous mode is usually used by network monitoring tools and transparent bridges (and, frequently, by network crackers trying to snatch passwords, or other data they're normally not able to see, off the wire).

5.2.6.5 Runt

A packet that is below the minimum size for a given protocol. With Ethernet, a runt is a frame shorter than the minimum legal length of 64 bytes (at Data Link). Runt packets are most likely the result of a collision, a faulty device on the network, or software gone awry.

5.2.6.6 Jabber

Jabber is a blanket term for a device that is behaving improperly in terms of electrical signalling on a network. In Ethernet this is Very Bad, because Ethernet uses electrical signal levels to determine whether the network is available for transmission. A jabbering device can cause the entire network to halt because all other devices think it is busy. Typically a jabber error results from a bad network interface card in a machine on the network. In bizarre circumstances outside interference might cause it. These are very hard problems to trace with layman tools.

5.2.6.7 Jam

When a workstation receives a collision, and it is transmitting, it puts out a jam so all other stations will see the collision also. When a repeater detects a collision on one port, it puts out a jam on all other ports, causing a collision to occur on those lines that are transmitting, and causing any non-transmitting stations to wait to transmit.

5.2.6.8 Broadcast storm

Basically it describes a condition where devices on the network are generating traffic that by its nature causes the generation of even more traffic. The inevitable result is a huge degradation of performance or complete loss of the network as the devices continue to generate more and more traffic. This can be related to the physical transmission or to very high level protocols.

To recognize a broadcast storm you have to be aware of the potential for it beforehand and be looking for it, because in a true broadcast storm you will probably be unable to access the network. This can change dramatically for a higher level protocol. NFS contention can result in a dramatic DROP in Ethernet traffic, yet no one will have access to resources.

5.2.6.9 Alignment Error

A received frame that does not contain an integer number of octets and contains a frame check sequence validation error. A frame in which the number of bits received is not an integer multiple of 8 and has a FCS (Frame Check Sequence) error.

5.3 Token Ring

Token Ring			
Size	Topology	Cable	Transmission
LAN	Physical Ring or Logical Ring, Physical Star	UTP STP Fiber	Packet Switching, using Token Passing

Token ring is the IEEE 802.5 standard that connects computers together in a closed ring. Devices on the ring cannot transmit data until permission is received from the network in the form of an electronic 'token'.

Token Ring is single access, meaning there is only one token. Thus, at any given time only one station is able to use the LAN. Since there is no such thing as a collision, all 4 or 16 Mbps can be used for frame transmission. This means that Ethernet with a practical utilization level of around 40% provides the same theoretically amount of bandwidth as 4Mbps token ring and 25% the bandwidth of 16Mbps token ring.

In terms of how much user data can theoretically be passed on the network per frame, the ratio between overhead and MTU is the best measure. For Ethernet it is 26/1518 = 0.017, for token ring it is 21/18000=.0012. This means that in the best case scenario, token ring transmits frames with one tenth the overhead associated with a similar transfer of Ethernet.

5.3.1 Token Ring Operation

A token ring network uses a special frame called a token that rotates around the ring when no stations are actively sending information. When a station wants to transmit on the ring, it must capture this token frame. The owner of the token is the only station that can transmit on the ring, unlike the Ethernet topology where any station can transmit at any time. Once a station captures the token, it changes the token into a frame format so data can be sent.

As the data traverses the ring, it passes through each station on the way to the destination station. Each station receives the frame and regenerates and repeats the frame onto the ring. As each station repeats the frame, it performs error checks on the information within the frame. If an error is found, a special bit in the frame called the Error Detection bit is set so other stations will not report the same error.

Once the data arrives at the destination station, the frame is copied to the destination's token ring card buffer memory. The destination station repeats the frame onto the ring, changing two series of bits on the frame. These bits, called the Address Recognized Indicator (ARI) and the Frame Copied Indicator (FCI), determines if the destination station had seen the frame and has had ample buffer space available to copy the frame into memory. If the frame is not copied into memory, it is the responsibility of the sending station to re-send the frame.

The frame continues around the ring, arriving back at the source station who recognizes the sending address as it's own. The frame is then stripped from the ring, and the source station sends a free token downstream.

5.3.2 Token Ring versus Ethernet

There is no 'converter' that allows an Ethernet network and Token Ring network to communicate between each other. A conversion process must occur between the two topologies, since they both use different signalling types, frame structures, and frame sizes.

There are two methods to accomplish this 'conversion'; bridging, and routing.

5.3.2.1 Bridging

Bridging is a method of communicating between devices at OSI layer 2, the data link layer. A bridge connects two networks together and acts as a traffic director. If traffic is destined to the other network, the bridge allows the traffic to pass. If the traffic is local to a single network, the bridge does not pass the traffic unnecessarily to the other connected network.

The bridge makes this determination based on the Media Access Control (MAC) address of the workstations on the network. The bridge keeps an updated list of everyone active on the network, and uses this list to direct traffic from one network to another.

This method of operation makes the network appear as a single logical network, since the only separation of traffic from one network to another is done at the MAC address level.

There are many bridge manufacturers and bridge types on the market. The newest version of this bridging technology is called a DLC Switch or LAN Switch. These switches have a much higher port density than the older two or three port bridges, allowing for much more flexibility and network segmentation.

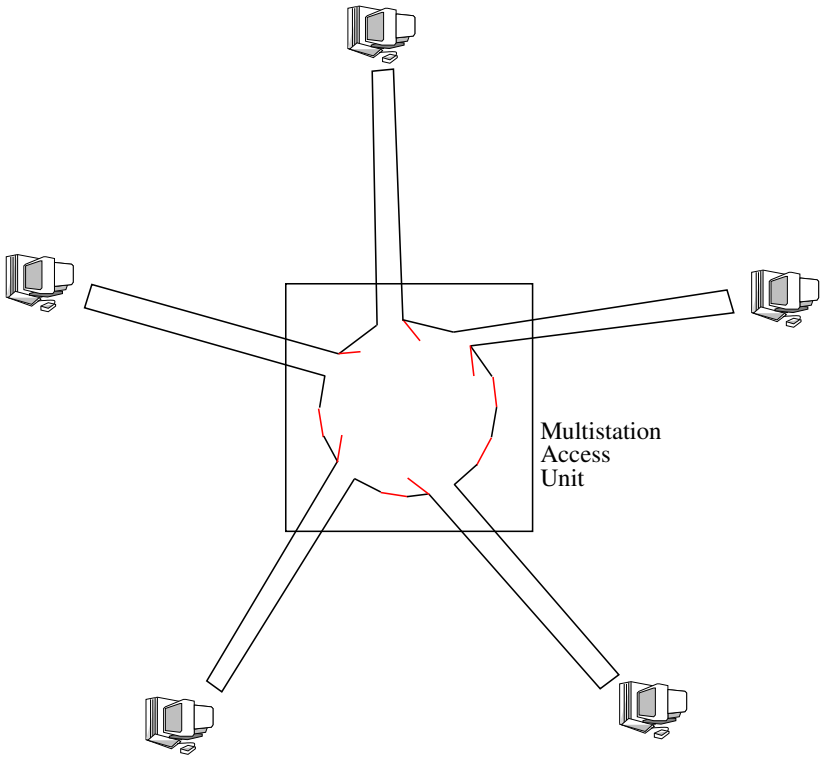


Figure 5.2: Token Ring, logical ring and physical star.

5.3.2.2 Routing

The second method of 'converting' from Ethernet to Token Ring is called routing. Routing occurs at OSI layer 3, and separates physical networks into separate logical networks. This differentiates routing from bridging, since bridging maintains a single logical network.

In a routed network, the sending workstation determines if outgoing traffic is local or remote. If the traffic belongs to another network, the originating station sends the frame directly to the router for further processing.

Upon receiving the frame from the source workstation, the router examines the frame for the destination address. The router maintains a routing table which is used to determine the final destination of the data packet through the router.

Routing is the most common method of connecting Ethernet networks to Token Ring networks in most organizations. Most network operating systems have routing capabilities built into the servers. By placing a token ring and Ethernet card into a Novell Netware 3.x/4.x or Windows NT v4.x server, the two topologies can communicate between each other.

One caveat; some protocols are unrouteable. A good example is Microsoft's NETBEUI, which has no OSI layer 3 network address and therefore cannot be routed. Protocols which cannot be routed must be bridged between physical networks.

5.3.3 Token Ring Physical Layer

Token ring connectivity requires three separate physical entities; a Multi-station Access Unit (MAU), a token ring lobe cable, and a token ring adapter card.

A Multi-station Access Unit (MAU or MSAU) is a hub-like device that connects to all token ring stations. Although the token ring stations are attached to the MAU in a physical star configuration, a true ring is maintained inside the MAU, as shown in Figure 5.2.

Unlike an Ethernet hub, a MAU consists of physical or electronic relays which keep each station in a loopback state until a voltage is sent from the station to the MAU. Since this voltage does not affect data communications, it is referred to as a 'phantom' voltage. Once this phantom voltage is received by the MAU, a relay is activated that inserts the token ring station onto the ring.

MAUs are connected together with Ring In/Ring Out (RI/RO) cables. To maintain a true ring, both the RI and the RO ports must be connected from one MAU to the other.

A token ring adapter card is the physical interface that a station uses to connect to a token ring network. There are token ring adapter cards for almost every computer bus type.

There are three major physical token ring cabling systems; Shielded Twisted Pair (STP), Unshielded Twisted Pair (UTP), and optic fiber.

Unlike Ethernet stations, token ring stations cannot be directly attached with a cross-over cable. Because of the process required for inserting into a ring, a loopback process must complete and phantom voltage must exist on a wire for a relay to open. A MAU must be used to directly connect two workstations.

In token ring networking, distance requirements are different from vendor to vendor. In general terms, the recommended standard distance between stations for Type 1 cabling is approximately 300 meters, and the recommended standard distance between stations for UTP cabling is about 150 meters.

Token ring distances are computed as the distance between repeaters. In a token ring network, each Network Interface Card (NIC) is a repeater. Therefore, the length between stations cannot exceed the cable lengths listed above.

Some manufacturers use 'active' MAUs which can regenerate the token ring signal and act as a repeater. In these cases, the distances between the token ring workstations and the MAUs can be much larger than many 'passive' MAUs. Many active MAUs have other network management features such as SNMP capabilities and auto-station removal for stations inserting at the incorrect speeds.

5.3.4 Token Ring Data Link Layer

5.3.4.1 MAC frame

A Media Access Control (MAC) frame is used for management of the token ring network. MAC frames do not traverse bridges or routers, since they carry ring management information for a single specific ring.

The MAC frame has this format:

SD	AC	FC	DA	SA	Data	FCS	ED	FS
1 byte	1 byte	1 byte	6 bytes	6 bytes	≥ 0	4 bytes	1 byte	2 bytes

- Starting Delimiter (SD): a single octet that consists of electrical signals that cannot appear elsewhere in the frame.
- Access Control (AC): includes priority and reservation bits used to set network priorities. It also includes a monitor bit, used for network management. A token bit indicates whether the frame is a token or a data frame.
- Frame Control (FC): The frame control field consists of eight bits, coded as TT00AAAA. The Frame Type bits (T) indicate the frame type. Bits 2 and 3 are reserved, and are always zero. Bits four through eight are Attention Codes which provide the token ring adapter with incoming MAC information that can be copied to a special Express Buffer in the token ring adapter.
- Destination Address (DA): The Destination Address specifies which station is to receive the frame. The Destination Address can be sent to a specific station, or a group of stations.
- Source Address (SA): The Source Address is the MAC address of the sending station.
- Data: A MAC frame data field contains token ring management information, and a non-MAC (LLC) data field contains user data.

- Frame Check Sequence (FCS): A 32 bit Cyclical Redundancy Check (CRC) is performed on the frame data to provide an integrity check of the frame data. As each station copies the frame, the CRC is computed and compared with the value in the FCS frame to verify that the frame data is correct.
- Ending Delimiter (ED): signals the end of the frame. This field includes two control bits. The intermediate bit indicates whether this is an intermediate or the final frame in a transmission. The error bit is set by any device that detects an error, such as in the FCS.
- Frame Status (FS): The Frame Status field provides information for the sending station regarding the status of the frame as it circulates the ring. The Frame Status field is coded as AF00AF00. The bits of the Frame Status field are duplicated, since this field does not fall under the CRC checking of the Frame Check Sequence bytes. The Address Recognized Indicator (ARI) is set to 1 by the destination station if the destination station recognizes the frame. The Frame Copied Indicator (FCI) is set to 1 if the destination station was able to copy the frame into the local adapter buffer memory.

5.3.4.2 LLC frames

A Logical Link Control (LLC) frame is used to transfer data between stations.

LLC frames have the same frame structure as MAC frames, except frame type bits of 01 are used in the Frame Control (FC) byte.

5.3.5 Monitors

Devices are either active monitors or standby monitors. There can only be a single active monitor on a physical token ring. Any station on the ring can assume the role of Active Monitor. All other stations on the ring are standby monitors.

The Active Monitor provides many functions on a token ring network:

- The Active Monitor is responsible for master clocking on the token ring network and the lower level management of the token ring network.
- The Active Monitor inserts a 24-bit propagation delay to prevent the end of a frame from wrapping onto the beginning of the frame.
- The Active Monitor confirms that a data frame or good token is received every 10 milliseconds. This timer sets the maximum possible frame size on a token ring network to 4048 bytes on a 4 megabit ring, and 17,997 bytes on a 16 megabit ring.
- The Active Monitor removes circulating frames from the ring. As a frame passes the Active Monitor, a special bit called a monitor count bit is set. If the monitor count bit is set, the Active Monitor assumes the original sender of the frame was unable to remove the frame from the ring. The Active Monitor purges this frame, and sends a Token Error Soft Error to the Ring Error Monitor.

If the Active Monitor is removed from the ring or no longer performs the Active Monitor functions, one of the Standby Monitors on the ring will take over as Active Monitor.

5.3.6 Reasons for Token Ring's lack of popularity

- It was developed as an IBM technology. Although Token Ring technology is now offered by great many vendors, many in the user community perceive it as proprietary.
- Ethernet is simple, reliable, and effective for the majority of networks, and at the same time, cost significantly less than Token Ring.
- TCP/IP has traditionally been wed to Ethernet. Growing industry demand for TCP/IP has accompanied a recent surge in the Ethernet popularity.

Nevertheless, Token Ring is an effective physical layer technology with features that make it preferable under some circumstances.

5.4 ISDN

ISDN			
Size	Topology	Cable	Transmission
WAN	Phone network Mesh Bus after NTU	Copper telephone lines	Circuit Switching

ISDN stands for “Integrated Services Digital Networks”, and it’s a ITU-T (formerly CCITT) term for a relatively new telecommunications service package. ISDN is basically the telephone network turned all-digital end to end, using existing switches and wiring (for the most part) upgraded so that the basic “call” is a 64 Kbps end-to-end channel, with bit-diddling as needed (but not when not needed!). Packet and maybe frame modes are thrown in for good measure, too, in some places. It’s offered by local telephone companies, but most readily in Australia, Western Europe, Japan, Singapore, and portions of the USA, and with other portions of USA somewhat more behind. In France, ISDN is known as "RNIS".

5.4.1 ISDN network connection

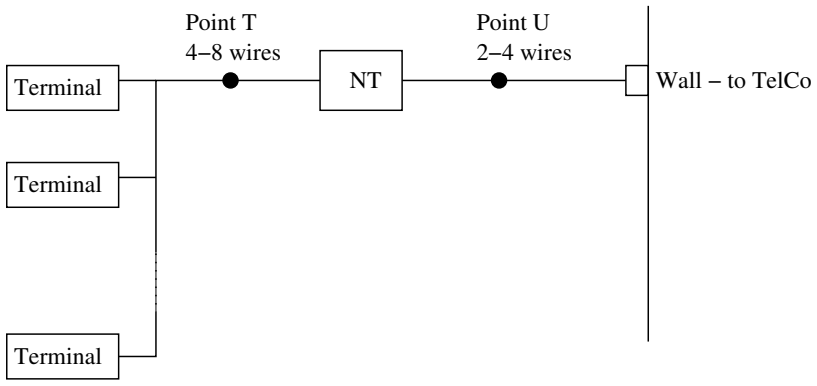
A Basic Rate Interface (BRI) is two 64K bearer (“B”) channels and a single delta (“D”) channel. The B channels are used for voice or data, and the D channel is used for signalling and/or X.25 packet networking. This is the variety most likely to be found in residential service.

Another flavor of ISDN is Primary Rate Interface (PRI). Inside North America and Japan, this consists of 24 channels, usually divided into 23 B channels and 1 D channel, and runs over the same physical interface as T1. Outside of these areas the PRI has 31 user channels, usually divided into 30 B channels and 1 D channel and is based on the E1 interface. It is typically used for connections such as one between a PBX (private branch exchange, a telephone exchange operated by the customer of a telephone company) and a CO (central office, of the telephone company) or IXC (inter exchange carrier, a long distance telephone company).

An ISDN BRI U-Loop is 2 conductors from the CO (telephone company central office) to the customer premises. Its maximum length may be 5.5 km. The equipment on both sides of the U loop has to be carefully designed to deal with the long length of the U loop and the noisy environment it operates in.

At the customer premises the U-loop is terminated by an NT1 (network termination 1) device. The NT1 drives an S/T-bus which is usually 4 wires, but in some cases it may be 6 or 8 wires.

This picture shows what a residential ISDN connection looks like.



The T bus is a multi-point bus in this configuration. It is sometimes called the passive bus because there are no repeaters on the line between the NT1 and the devices. It can be implemented using the same cable and connectors as is 10 base T Ethernet. There may be up to 8 devices on the S/T bus. The bus may be formed with splitters and T connectors - it is a bus, not a star. The D channel is used to control the attachment of the one to eight devices to the two B channels. No two devices attach to the same B channel at the same time.

In this configuration, the major function of the NT is to allow more than one device to have access to the 2 B channels provided by the ISDN BRI. For instance, you may have an ISDN telephone, an ISDN fax and an ISDN computer interface attached to the BRI. Each device can listen for calls and only connect to a B channel when it identifies a message requesting a service it can provide.

The NT1 only implements part of the channel sharing scheme; the other devices participate as well, and the communication protocol used by the NT1 and the other devices is an integral part of the scheme. The NT1 also performs other functions; it translates the bit encoding scheme used on the lines between it and the telephone company (the U loop) to the encoding used between it and the devices. These schemes are different because the device to NT encoding was designed to enable channel sharing whereas the NT to exchange encoding was designed to allow transmission across long distances.

The ISDN pairs are the same wires as used for regular telephone service. If you became an ISDN user at home, the same wire pair that now provides your telephone service would be used to provide ISDN (assuming you no longer have the regular line).

Most of the lines do not require any special conditioning. If a line has load coils on it they must be removed, but load coils are usually only found on existing lines that are 15,000 feet or longer. As to lines with bridge taps, the 2B1Q line transmission scheme (not to be confused with 2B + D channelization) is tolerant of a certain amount of bridge taps and, therefore it is only a minimal subset of existing lines (lines with bridge taps whose total length is greater than 3000 feet for the bridge taps) that would require special "deconditioning."

Plain old telephone service is transmitted between the central office to your home or office telephone set (or modem, or fax) in analog form. At the central office, the analog signal is converted to a series of digital samples at a rate of 8000 samples per second. Each sample is seven or eight bits in length. As the signals for a telephone call move around the central office, or between central offices, they are transmitted in digital form. Thus, a telephone call consumes a transmission bandwidth of either 56 or 64 kilobits per second. The theoretical (Nyquist) limit for the frequency response of a signal sampled 8000 times per second is 4kHz. However, due to various losses in the telephone system, the frequency response of an ordinary telephone call is usually quoted as 3.1kHz. Ordinary modem-based data transmission uses schemes for encoding data in an analog signal so it fits in this 3.1kHz bandwidth. 14.4Kbps is a commonly available transmission rate at the high end of the scale. With this transmission rate, over three-quarters of the bit rate handled by the central office is wasted.

Notice that in telephony, 64 Kbps means 64000 bits per second, whereas in computer engineering 64k bytes typically means 65536 bytes.

ISDN brings the digital signal all the way to your home or desktop. With ISDN, you can place a data call which uses all 56Kbps or 64Kbps, because there is no need to convert the signal to analog in your modem and back to digital at the central office. The availability of the full bandwidth presents some interesting technological opportunities:

- transmission of high-fidelity compressed audio
- transmission of encrypted audio
- transmission of lots of data
- transmission of other compressed signals, such as video

Basic-rate ISDN (BRI) offers two channels of this service. In BRI, the connection between your site and the central office offers 64Kbps bidirectionally on each channel. Each of these channels may be used for a voice call, for circuit-switched data, or for X.25 packet switched data. Thus, the existing POTS circuit [POTS: Plain Old Telephone Service, i.e. traditional analog telephony] can be conditioned to carry two calls at the same time.

Incidentally, ISDN brings another interesting service to your home or desktop: a highly reliable 8000Hz clock signal. In most cases, the central office switches, long-distance carriers, and ISDN terminal equipment all operate with exactly the same clock frequency. In a real-time communications environment (like a voice phone call) this means that there's no need to compensate for differences between the sampling rates at each end of the call.

One of the other features is that instead of the CO sending an AC ring signal to activate your bell, it sends a digital packet that tells WHO is calling (if available), WHAT TYPE of call (speech, data communications), the NUMBER DIALED (maybe one of your aliases) and some other stuff. Your equipment can then analyze this stuff and make an "intelligent" decision what to do with it. For example, a phone (with speech-only capacity) would completely ignore a data call while a Terminal Adapter (ISDN "modem") or a phone with built-in data communication functions would respond to it. If you have several "aliases" tied to your line, you can program certain phones to answer calls for certain numbers only. Data calls contain baud rate and protocol information within the setup signal so that the connection is virtually instantaneous (no messing around with trying different carriers until both ends match).

Broadband ISDN refers to services that require channel rates greater than a single primary rate channel. While this does not specifically imply any particular technology, ATM will be used as the switching infrastructure for B-ISDN services.

5.5 FDDI

FDDI			
Size	Topology	Cable	Transmission
LAN backbone MAN WAN	Physical Tree Logical Ring ($\times 2$)	Fiber	Packet Switching, using Token Passing

Earlier types of LANs, such as Ethernet and Token Ring , operate at bit rates ranging from 1 to 16 Mbps. Two examples of newer and high speed LANs which accommodate up-to-date demands are DQDB (Distributed Queue Dual Bus) and FDDI (Fiber Distributed Data Interface).

In addition to asynchronous data (which is generated at random time intervals), FDDI rings can also support the transmission of synchronous data (for example, digitized voice).

FDDI Physical Parameters:

- Media: 1300nm, optical fibers.
- Transmission Method: Baseband.
- Data Rate: 100Mbps.
- Topology: physical ring of trees, logical ring (see Figure 5.3).
- Maximum distance between adjacent stations: 2 km.
- Total max. ring length: 100 km.
- Max. number of attached stations: 1000.

5.5.1 Network Configuration

FDDI uses two counter rotating rings in the same way as a Token Ring LAN to enhance reliability: one is referred to as the Primary Ring and the other as the Secondary Ring. The secondary ring can be used either

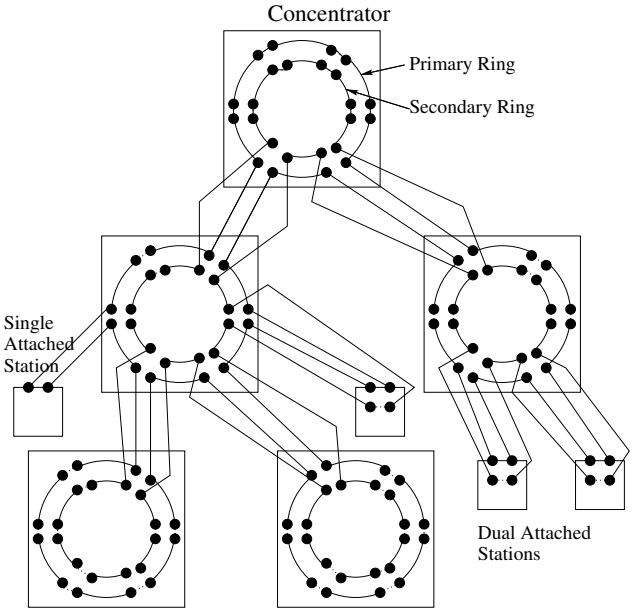


Figure 5.3: FDDI Topology: Physical tree, logical ring

as an additional transmission path or purely as a backup in the event of a break occurring in the primary ring.

There are 4 types of stations (DTEs or Concentrators):

1. Dual Attached Station (DAS), which is connected to both rings.
2. Single Attached Station (SAS), which is attached only to the primary ring.
3. Dual Attached Concentrator (DAC), which is connected to both rings and provides connection for additional stations and concentrators. It is actually the root of a tree.
4. Single Attached Concentrator (SAC), which is connected only to the primary ring (through a tree).

In practice, most user stations are attached to the ring via wiring concentrators, since then only a single pair of fibers is needed and the connection cost is lower. The basic fiber is dual core with polarized duplex connectors at each end. This means that each end of the cable has a different physical key so that it can only be connected into a matching socket (to prevent faulty interchanging of wires which can cause a total break- down of the network). Special coupling units are used to isolate (bypass) a station when its power is lost (either active or passive fiber devices). Stations detecting a cable break will go into wrap mode (use the secondary ring as backup so both rings are connected to form a single ring) - See Figure 5.4.

5.5.2 Physical Interface

As opposed to a basic Token Ring network, in which at any instant there is a single active ring monitor which supplies the master clock for the ring, in FDDI this approach is not suitable because of the high data rates. Instead, each ring interface has its own local clock, and outgoing data is transmitted using this clock.

All data to be transmitted is encoded prior to transmission using a 4 of 5 group code. This means that for each 4 bits of data a corresponding 5 bit code word or symbol is generated by the encoder. Some of these symbols (combinations) are used for link control functions (such as indicating the start and end of each transmitted frame or token). In general, the meaning and use of FDDI frame (or token) fields is the same as with the basic Token Ring, but because of the use of symbols rather than bits there are some differences in the structure of each field.

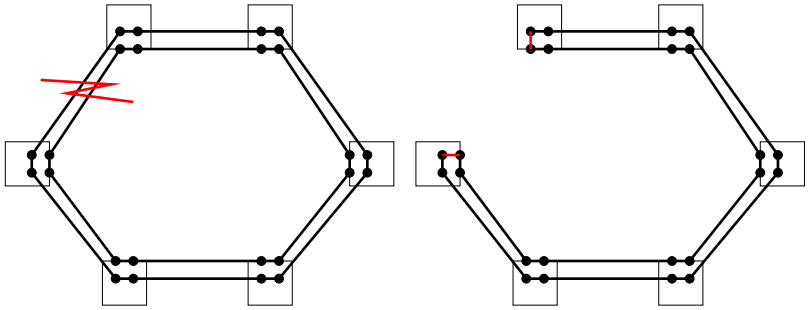


Figure 5.4: FDDI ring entering wrap mode.

5.5.3 Ring Operation

FDDI's ring operation is basically very similar to the Token Ring early release operation.

The main steps that take place in a normal frame transmission are as following:

1. Sending station waits for token.
2. Sending station captures and strips token, and then transmits frames.
3. Sending station issues token at the end of transmission.
4. Destination station copies the transmitted frame and sets the A&C bits (Address Recognized and Frame Copied indicators).
5. Sending station removes the data from the ring by stripping the sent (and acknowledged) frame.
6. The first bytes of the frame are not stripped, and continue to circulate on the ring (as a fragment). Each repeating station strips one byte from the fragment, and a transmitting station completely strips it.

Unlike the basic Token Ring, which is based on the use of priority and reservation bits, the priority operation of the FDDI ring uses a principle that is based on a parameter known as the Token Rotation Time (TRT). The TRT is the time that has expired since a station last received the token. It thus includes the time taken by this station to transmit any waiting frames, as well as the time taken by other stations in the ring for this rotation of the token. Clearly, if the ring is lightly loaded, then the TRT is short. As the loading on the ring increases, so the TRT measured by each station increases. Thus, the TRT is a measure of the total ring loading.

An example of the importance of the use of good priority mechanism on FDDI is the need to provide high priority to the transmission of synchronous data, since late arrival of such data would render it useless.

5.6 ATM

ATM			
Size	Topology	Cable	Transmission
LAN (usually backbone) MAN WAN	Star in LAN Mesh in WAN	Fiber UTP STP	Packet (Cell) Switching, with virtual channels

ATM was developed because of developing trends in the networking field. The most important parameter is the emergence of a large number of communication services with different, sometimes yet unknown requirements. In this information age, customers are requesting an ever increasing number of new services. The most famous communication services to appear in the future are HDTV(High Definition TV), video conferencing, high speed data transfer, video telephony, video library, home education and video on demand.

This large span of requirements introduces the need for one universal network which is flexible enough to provide all of these services in the same way. Two other parameters are the fast evolution of the semiconductor and optical technology and the evolution in system concept ideas - the shift of superfluous transport functions to the edge of the network.

Both the need for a flexible network and the progress in technology and system concepts led to the definition of the Asynchronous Transfer Mode (ATM) principle.

The networks of today are very specialized and suffer from a large number of disadvantages:

- Service Dependence: Each network is only capable of transporting one specific service.
- Inflexibility: Advances in audio, video and speech coding and compression algorithms and progress in VLSI technology influence the bit rate generated by a certain service and thus change the service requirements for the network. In the future new services with unknown requirements will appear. A specialized network has great difficulties in adapting to new services requirements.
- Inefficiency: The internal available resources are used inefficiently. Resources which are available in one network cannot be made available to other networks.

The ideal network in the future must be flexible. The most flexible network in terms of bandwidth requirements and the most efficient in terms of resource usage, is a network based on the concept of packet switching. Any bandwidth can be transported over a packet switching network and the resources are only used when useful information has to be transported. The basic idea behind the concept changes is the fact that functions must not be repeated in the network several times if the required service can still be guaranteed when these functions are only implemented at the boundary of the network.

5.6.1 Principles of ATM

ATM is considered a packet oriented transfer mode based on:

- asynchronous time division multiplexing
- the use of fixed length cells

Each cell consist of an information field and a header.

- The header is used to identify cells belonging to the same virtual channel and to perform the appropriate routing. To guarantee a fast processing in the network, the ATM header has very limited function. Its main function is the identification of the virtual connection by an identifier which is selected at call set up and guarantees a proper routing of each packet. In addition it allows an easy multiplexing of different virtual connections over a single link.
- The information field length is relatively small, in order to reduce the internal buffers in the switching node, and to limit the queuing delays in those buffers - small buffers guarantee a small delay and a small delay jitter as required in real time systems. The information field of ATM cells is carried transparently through the network. No processing is performed on it inside the network. All services (voice, video, data) can be transported via ATM , including connectionless services.

ATM is connection oriented. Before information is transferred from the terminal to the network, a logical/virtual connection is set. The header values are assigned to each section of a connection for the complete duration of the connection, and translated when switched from one section to another. Signalling and user information are carried on separate virtual channels. Two sorts of connections are possible:

- Virtual Channel Connections VCC
- Virtual Path Connections VPC

When switching or multiplexing on cells is to be performed, it must first be done on VPC ,then on the VCC.

5.6.1.1 Virtual Channels

This function is performed by a header sub field - VCI. Since the ATM network is connection oriented each connection is characterized by a VCI which is assigned at call set up. A VCI has only a local significance on the link between ATM node and will be translated in the ATM nodes. When the connection is released , the VCI values on the involved links will be released and can be reused by other connections. An advantage of this VCI principle is the use of multiple VCI values for multicomponent services. For instance video telephony can be composed of 3 components: voice , video and data each of which will be transported over a separate VCI. This allows the network to add or remove components during the connection. For instance, the video telephony service can start with voice only and the video can be added later.

5.6.1.2 Virtual Path

The network has to support semi-permanent connections, which have to transport a large number of simultaneous connections. This concept is known as virtual path.

5.6.2 ATM Facilities

As ATM is connection oriented, connections are established either semi-permanently, or for the duration of a call, in case of switched services. This establishment includes the allocation of a VCI (Virtual Channel Identifier)and/or VPI (Virtual Path Identifier), and also the allocation of the required resources on the user access and inside the network. These resources are expressed in terms of throughput and Quality of Service. They may be negotiated between user and network for switched connection during the call set up phase

5.6.2.1 ATM Cell Identifiers

ATM cell identifiers are:

- Virtual Path Identifier
- Virtual Channel Identifiers
- Payload Type Identifiers

They support recognition of an ATM cell on a physical transmission medium. Recognition of a cell is a basis for all further operations. VPI and VCI are unique for cells belonging to the same virtual connection on a shared transmission medium. As such they are limited resources. Within a particular virtual circuit, cells may be further distinguished by their PTI, which cannot be allocated freely but depends on the type of payload carried by the cell. This field indicates whether the cell is carrying user information to be delivered transparently through the network or special network information. In case the field indicates network information, part of the information field indicates the type of network control whereas the remaining part of information field may be processed inside the network.

Throughput Bandwidth has to be reserved in the network for each virtual connection. ATM offers the possibility to realize resources saving in the total bandwidth needed when multiplexing traffic of many variable Bit Rate connections. The amount which can be saved depends heavily on the number of multiplexed connections, on the burstiness of the traffic they carry, on the correlation between them and on the quality of service they require.

5.6.2.2 Quality of Service

The quality of service of a connection relates to the cell loss, the delay and the delay variation incurred by the cells belonging to that connection in an ATM network. For ATM, the quality of service of a connection is closely linked to the bandwidth it uses. When providing limited physical resources using more bandwidth increases the cell loss, the delay, and the delay variation incurred, i. e. decreases the QOS for cells of all connections which share those resources.

5.6.2.3 Usage Parameter Control

In ATM there is no physical limitation on the user access rate to the physical transmission medium, apart from the physical cell rate on the medium itself. Multiplexing equipment will do its utmost to avoid cell loss to offer the highest possible throughput whatever the user chooses to send. As virtual connections share physical resources, transmission media and buffer space, unforeseen excessive occupation of resources by one user may impair traffic for other users. Throughput must be monitored at the user - network interface by a Usage Parameter Control function in the network to ensure that a negotiated contract per VCC or VPC between network and subscriber is respected. It is very important that the traffic parameters which are selected for this purpose can be monitored in real time at the arrival of each cell.

5.6.2.4 Flow Control

In principle, no flow control will be applied to information streams at the ATM layer of the network. In some cases it will be necessary to be able to control the flow of traffic on ATM connections from a terminal to the network. In order to cope with this a GFC (general flow control) mechanism may be used. This function is supported by a specific field in the ATM cell header. Two sets of procedure are used:

- Uncontrolled Transmission - for the use of point to point configuration.
- Controlled Transmission - can be used in both point to point and shared medium configuration.

Another principle is no error protection on link by link basis. If a link in the connection, either the user to network link or the internal links between the network nodes, introduces an error during the transmission or is temporarily overloaded thereby causing the loss of packets, no special action will be taken on that link to correct this error (= no requesting for retransmission).This error protection can be omitted since the links in the network have a very high quality

5.6.2.5 Signalling

The negotiation between the user and the network with respect to the resources is performed over a separate signalling virtual channel. The signalling protocol to be used over the signalling virtual channel is an enhancement of those used in ISDN signalling.

5.6.3 ATM - The Layered Model

The OSI model is very famous and used to model all sorts of communication systems. We can model the ATM with the same hierarchical architecture - however only the lower layers are used.

The following relations can be found:

- The Physical layer is more or less equivalent to Layer 1 of OSI model, and mainly perform functions on the bit level.
- The ATM layer can be located mainly at the lower edge of the layer 2 of the OSI model.
- The adaptation layer performs the adaptation of higher layer protocols, be it signalling or user information, to the fixed ATM cells.

These layers can then further be divided into sub-layers.

- PM - Physical Medium Sublayer: This sublayer is responsible for the correct transmission and reception of bits on the appropriate physical medium. At the lowest level the functions that are performed are medium dependent: optical, electrical... In addition this sublayer must guarantee a proper bit timing reconstruction at the receiver. Therefore the transmitting peer will be responsible for the insertion of the required bit timing information and line coding.

- Transmission Convergence Sublayer: In this sublayer bits are already recognized, as they come from the PM sublayer. This sublayer performs the following functions:

  - Adaptation to the transmission system used
  - Generation of the HEC (Header Error Check) of each cell at the transmitter, and its verification at the receiver
  - Cell delineation - the mechanism to perform cell delineation is based on the HEC. If a correct HEC is recognized for a number of consecutive cells it is assumed that the correct cell boundary is found. To avoid erroneous cell delineation on user information, the information field of each cell is scrambled at the transmitting side and de-scrambled at the receiving side. This ensures that the probability of finding a correct HEC in the information field is very low
  - Once the cell delineation has been found an adaptive mechanism uses the HEC for correction or detection of cell header errors. Isolated single bit errors are corrected.
  - Cell uncoupling - the sublayer ensures insertion and suppression of unassigned cells to adapt the useful rate to the available payload of the transmission system

- ATM Layer The following main functions are performed by the layer:

  - The multiplexing and demultiplexing of cells of different connections into a single cell stream
  - A translation of cell identifiers, which is required in most cases when switching a cell from one physical link to another in an ATM switch or cross connect. This translation can be performed either on the VCI or VPI separately, or on both simultaneously.
  - Providing the user of a VCC or VPC with one QOS class out of a number of Classes supported by the network.
  - Management functions: the header of user information cells provides for a congestion indication and an ATM user to ATM user indication.
  - Extraction (addition) of the cell header before (after) the cell is being delivered to (from) the adaptation layer
  - Implementation of flow control mechanism on the user network interface.

- ATM Adaptation Layer This layer enhances the service provided by the ATM layer to a level required by the next higher layer. It performs the functions for the user, control and management planes and supports the mapping between the ATM layer and the next higher layer. The functions performed in the AAL depend on the higher layer requirements. The AAL layer is divided into two sub-layers:

  - SAR - the segmentation and reassembly sublayer The main purpose of the SAR sublayer is segmentation of higher layer information into a size suitable for the payload of the consecutive ATM cells of a virtual connection, and the inverse operation, reassembly of contents of the cells of a virtual connection into data units to be delivered to the higher layer.
  - CS - the convergence sublayer This sublayer performs functions like message identification, time/clock recovery etc. AAL Service Data Units (SDU) are transported from one AAL Service Access Point to one or more access points through the ATM network. The AAL users will have the capability to select a given AAL - SAP associated with the QOS required to transport the SDU. Up to now four AALS have been defined-one for each class of service.

5.6.4 Classes of ATM Services

The services which will be transported over the ATM layer are classified in four classes, each of which has its own specific requirements toward the AAL. The services are classified according to three basic parameters:

1. Time relation between source and destination: For real time applications like phone conversation, a time relation is required. Information transfer between computers does not require a time relation.
2. Bit Rate Some services have a constant bit rate, others have a variable bit rate.
3. Connection mode: connectionless or connection oriented

Four types of AAL protocols have been recommended up to now : AAL 1,AAL 2, AAL 3/4, AAL 5.

1. AAL 1 - Adaptation for constant bit rate services: Recommended for services such as digital voice and digital video. It is used for applications that are sensitive for both cell loss and delay. Constant Bit Rate (CBR) services require information to be transferred between source and destination at a constant bit rate after virtual connection has been set up.
2. AAL 2 - Adaptation for variable bit rate services: This type AAL offers a transfer of information with a variable bit rate. In addition, timing information is transferred between source and destination. Since the source is generating a variable bit rate, it is possible that cells are not completely filled and that the filling level varies from cell to cell.
3. AAL 3/4 - Adaptation for data services: This AAL is recommended for transfer of data which is sensitive to loss, but not to delay. The AAL may be used for connection oriented as well as for connectionless services, since functions like routing and network addressing are performed on the network layer.
4. AAL 5 - Adaptation for data services: This AAL is recommended for high speed connection oriented data service. This AAL offers a service with less overhead and better error detection.

5.7 Gigabit Ethernet

Ethernet is the world's most pervasive networking technology. Gigabit Ethernet is one of the latest versions of Ethernet. It offers 1000 Mbps (1 Gbps) raw bandwidth, that is 100 times faster than the original Ethernet, yet is compatible with existing Ethernets, as it uses the same CSMA/CD and MAC protocols. When Gigabit Ethernet enters the market it will compete directly with ATM.

Ethernet is the world's most pervasive networking technology , since the 1970's. It is estimated that in 1996, 82% of all networking equipment shipped was Ethernet. In 1995 ,the Fast Ethernet Standard was approved by the IEEE. Fast Ethernet provided 10 times higher bandwidth, and other new features such as full-duplex operation, and auto-negotiation. This established Ethernet as a scalable technology. Now, with the emerging Gigabit Ethernet standard, it is expected to scale even further.

The Fast Ethernet standard was pushed by an industry consortium called the Fast Ethernet Alliance. A similar alliance, called the Gigabit Ethernet Alliance was formed by 11 companies in May 1996 , soon after IEEE announced the formation of the 802.3z Gigabit Ethernet Standards project. At last count, there were over 95 companies in the alliance from the networking, computer and integrated circuit industries.

The new Gigabit Ethernet standards will be fully compatible with existing Ethernet installations. It will retain Carrier Sense Multiple Access/ Collision Detection (CSMA/CD) as the access method. It will support full-duplex as well as half duplex modes of operation.

Initially, Gigabit Ethernet is expected to be deployed as a backbone in existing networks. It can be used to aggregate traffic between clients and "server farms", and for connecting Fast Ethernet switches. It can also be used for connecting workstations and servers for high - bandwidth applications such as medical imaging or CAD.

5.7.1 Physical Layer

The Physical Layer of Gigabit Ethernet uses a mixture of proven technologies from the original Ethernet and the ANSI X3T11 Fiber Channel Specification. Gigabit Ethernet supports a number of physical media types:

1000Base-X A set of media based on fiber optic cable. Three types of media are include in the 1000Base-X standard :

1000Base-SX 850 nm laser on multi mode fiber for a distance of about 300m.

1000Base-LX 1300 nm laser on single mode (3km) and multi mode fiber (500m).

1000Base-CX Short haul copper STP (Shielded Twisted Pair) cable limited to lengths of 25m.

1000Base-T 1000Base-T is a standard for Gigabit Ethernet over long haul copper UTP. This can be used in lengths of up to 100m, but requires 4 pairs of Category 5 UTP.

5.7.2 MAC Layer

The MAC Layer of Gigabit Ethernet uses the same CSMA/CD protocol as Ethernet. The maximum length of a cable segment used to connect stations is limited by the CSMA/CD protocol. If two stations simultaneously detect an idle medium and start transmitting, a collision occurs.

Ethernet has a minimum frame size of 64 bytes. The reason for having a minimum size frame is to prevent a station from completing the transmission of a frame before the first bit has reached the far end of the cable, where it may collide with another frame. Therefore, the minimum time to detect a collision is the time it takes for the signal to propagate from one end of the cable to the other. This minimum time is called the Slot Time. A more useful metric is Slot Size, the number of bytes that can be transmitted in one Slot Time. In Ethernet, the slot size is 64 bytes, the minimum frame length.

The maximum cable length permitted in Ethernet is 2.5 km (with a maximum of four repeaters on any path). As the bit rate increases, the sender transmits the frame faster. As a result, if the same frames sizes and cable lengths are maintained, then a station may transmit a frame too fast and not detect a collision at the other end of the cable. So, one of two things has to be done:

1. Keep the maximum cable length and increase the slot time (and therefore, minimum frame size) or,

2. keep the slot time same and decrease the maximum cable length.

In Fast (100Mbps) Ethernet, the maximum cable length is reduced to only 100 meters, leaving the minimum frame size and slot time intact.

Gigabit Ethernet maintains the minimum and maximum frame sizes of Ethernet. Since, Gigabit Ethernet is 10 times faster than Fast Ethernet, to maintain the same slot size, maximum cable length would have to be reduced to about 10 meters, which is not very useful. Instead, Gigabit Ethernet uses a bigger slot size of 512 bytes. To maintain compatibility with Ethernet, the minimum frame size is not increased, but the "carrier event" is extended. If the frame is shorter than 512 bytes, then it is padded with extension symbols. These are special symbols, which cannot occur in the payload. This process is called Carrier Extension.

5.7.2.1 Carrier Extension

For carrier extended frames, the non-data extension symbols are included in the "collision window", that is, the entire extended frame is considered for collision and dropped. However, the Frame Check Sequence (FCS) - the CRC - is calculated only on the original (without extension symbols) frame. The extension symbols are removed before the FCS is checked by the receiver. So the LLC (Logical Link Control) layer is not even aware of the carrier extension.

Carrier Extension is a simple solution, but it wastes bandwidth. Up to 448 padding bytes may be sent for small packets. This results in low throughput. In fact, for a large number of small packets, the throughput is only marginally better than Fast Ethernet.

5.7.2.2 Packet Bursting

Packet Bursting is an extension of Carrier Extension. Packet Bursting is "Carrier Extension plus a burst of packets". When a station has a number of packets to transmit, the first packet is padded to the slot time if necessary using carrier extension. Subsequent packets are transmitted back to back, with the minimum Inter-packet gap (IPG) until a burst timer (of 1500 bytes) expires. Packet Bursting substantially increases the throughput.

5.7.3 Gigabit Ethernet versus ATM

When ATM (Asynchronous Transfer Mode) was introduced, it offered 155 Mbps bandwidth, which was 1.5 times faster than Fast Ethernet. ATM was ideal for new applications demanding a lot of bandwidth, especially multimedia. Demand for ATM continues to grow for LANs as well as WANs.

On the one hand , proponents of ATM try to emulate Ethernet networks via LANE (LAN Emulation) and IPOA (IP over ATM). On the other, proponents of Ethernet/IP try to provide ATM functionality with RSVP (Resource Reservation Protocol) and RTSP (Real-time Streaming Transport Protocol). Evidently, both technologies have their desirable features, and advantages over the other. It appears that these seemingly divergent technologies are actually converging.

ATM was touted to be the seamless and scalable networking solution - to be used in LANs, backbones and WANs alike. However, that did not happen. And Ethernet, which was for a long time restricted to LANs alone, evolved into a scalable technology.

As Gigabit Ethernet products enter the market, both sides are gearing up for the battle. Currently, most installed workstations and personal computers do not have the capacity to use these high bandwidth networks. So, the imminent battle is for the backbones, the network connections between switches and servers in a large network.

ATM still has some advantages over Gigabit Ethernet :

- ATM has a head start over Gigabit Ethernet. Current products may not support Gigabit speeds, but faster versions are in the pipeline.

- ATM is better suited than Ethernet for applications such as video, because ATM has QOS (Quality of Service) and different services available such as CBR (constant bit rate) which are better for such applications. Though the IETF (Internet Engineering Task Force, the standards body for Internet protocols) is working on RSVP which aims to provide QOS on Ethernet, RSVP has it's limitations. It is a "best effort" protocol, that is , the network may acknowledge a QOS request but not deliver it. In ATM it is possible to guarantee QOS parameters such as maximum delay in delivery.

Gigabit Ethernet has its own strengths :

- The greatest strength is that it is Ethernet. Upgrading to Gigabit Ethernet is expected to be painless. All applications that work on Ethernet will work on Gigabit Ethernet. This is not the case with ATM. Running current applications on ATM requires some amount of translation between the application and the ATM layer, which means more overhead.

- Currently, the fastest ATM products available run at 622 Mbps. At 1000 Mbps, Gigabit Ethernet is almost twice as fast.

It is not clear whether any one technology will succeed over the other. It appears that sooner or later, ATM and Ethernet will complement each other and not compete.

5.7.3.1 What is next?

The 10 Gigabit Ethernet specification is being submitted for standardization. Ethernet is being proposed as a WAN protocol.

5.8 Wireless Networks

For some time now, companies and individuals have interconnected computers with local area networks (LANs). This allowed the ability to access and share data, applications and other services not resident on any one computer. The LAN user has at their disposal much more information, data and applications than they could otherwise store by themselves. In the past all local area networks were wired together and in a fixed location.

An increasing number of LAN users are becoming mobile. These mobile users require that they are connected to the network regardless of where they are because they want simultaneous access to the network. This makes the use of cables, or wired LANs, impractical if not impossible. Wireless LANs are very easy to install. There is no requirement for wiring every workstation and every room. This ease of installation makes wireless LANs inherently flexible. If a workstation must be moved, it can be done easily and without additional wiring, cable drops or reconfiguration of the network. Another advantage is its portability. If a company moves to a new location, the wireless system is much easier to move than ripping up all of the cables that a wired system would have snaked throughout the building. Most of these advantages also translate into monetary savings.

5.8.1 Physical Media

There are three media that can be used for transmission over wireless LANs: Infrared, radio frequency and microwave. In the United States the industrial, scientific, and medical radio frequency bands do not require licensing. This prompted most of the wireless LAN products to operate within these bands. In the U.S. radio frequency (RF) systems must implement spread spectrum technology. Wireless systems must also operate at low power.

5.8.1.1 Infrared

Infrared systems are simple in design and therefore inexpensive. They use the same signal frequencies used on fiber optic links. IR systems detect only the amplitude of the signal and so interference is greatly reduced. These systems are not bandwidth limited and thus can achieve transmission speeds greater than the other systems. Infrared transmission operates in the light spectrum and does not require a license to operate, another attractive feature. There are two conventional ways to set up an IR LAN. The infrared transmissions can be aimed. This gives a good range of a couple of kilometers and can be used outdoors. It also offers the highest bandwidth and throughput. The other way is to transmit omni-directionally and bounce the signals off of everything in every direction. This reduces coverage to 30 - 60 feet, but it is an area coverage. IR technology was initially very popular because it delivered high data rates and relatively cheap price. The drawbacks to IR systems are that the transmission spectrum is shared with the sun and other things such as fluorescent lights. If there is enough interference from other sources it can render the LAN useless. IR systems require an unobstructed line of sight (LOS). IR signals cannot penetrate opaque objects. This means that walls, dividers, curtains, or even fog can obstruct the signal.

5.8.1.2 Microwave

Microwave systems are by far the fewest on the market. They use narrow-band transmission with single frequency modulation and are set up mostly in the 5.8GHz band. They achieve higher throughput because they do not have the overhead involved with spread spectrum systems.

5.8.1.3 Radio

Radio frequency systems must use spread spectrum technology in the United States. This spread spectrum technology currently comes in two types: direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS).

5.8.1.3.1 Direct Sequence Spread Spectrum (DSSS) With direct sequence spread spectrum the transmission signal is spread over an allowed band (for example 25MHz). A random binary string is used to modulate the transmitted signal. This random string is called the spreading code. The data bits are mapped into a pattern of "chips" and mapped back into a bit at the destination. The number of chips that represent a bit is the spreading ratio. The higher the spreading ratio, the more the signal is resistant to interference. The lower the spreading ratio, the more bandwidth is available to the user.

Most products have a spreading ratio of less than 20 and the new IEEE 802.11 standard requires a spreading ratio of eleven. The transmitter and the receiver must be synchronized with the same spreading code. If orthogonal spreading codes are used then more than one LAN can share the same band. However, because DSSS systems use wide sub-channels, the number of co-located LANs is limited by the size of those sub-channels. Recovery is faster in DSSS systems because of the ability to spread the signal over a wider band.

5.8.1.3.2 Frequency Hopping Spread Spectrum (FHSS) This technique splits the band into many small sub-channels (1MHz). The signal then hops from sub-channel to sub-channel transmitting short bursts of data on each channel for a set period of time, called dwell time. The hopping sequence must be synchronized at the sender and the receiver or information is lost.

Frequency hopping is less susceptible to interference because the frequency is constantly shifting. This makes frequency hopping systems extremely difficult to intercept. This feature gives FH systems a high degree of security. In order to jam a frequency hopping system the whole band must be jammed. These features are very attractive to agencies involved with law enforcement or the military. Many FHSS LANs can be co-located if an orthogonal hopping sequence is used. Because the sub-channels are smaller than in DSSS, the number of co-located LANs can be greater with FHSS systems.

5.8.1.4 Media Issues

5.8.1.4.1 Multipath Interference caused by signals bouncing off of walls and other barriers and arriving at the receiver at different times is called multipath interference, and affects all the wireless media discussed above. FHSS inherently solves the multipath problem by simply hopping to other frequencies. Other systems use anti-multipath algorithms to avoid this interference.

A subset of the multipath problem is Rayleigh fading. This occurs when the difference in path length is arriving from different directions and is a multiple of half the wavelength. Rayleigh fading has the effect of completely canceling out the signal. IR is not effected by Rayleigh fading because the wavelengths used in IR are so small.

5.8.2 802.11

With more and more companies and individuals requiring portable and mobile computing the need for wireless local area networks continues to rise throughout the world. Because of this growth, IEEE formed a working group to develop a Medium Access Control (MAC) and Physical Layer (PHY) standard for wireless connectivity for stationary, portable, and mobile computers within a local area. This working group is IEEE 802.11.

5.8.2.1 802.11 Physical layer

The three PHY layers defined by the 802.11 standard are, confusingly, not inter-operable. They are: infrared (never implemented by anybody); Frequency Hopping Spread Spectrum (FHSS) radio; and Direct Sequence Spread Spectrum (DSSS) radio.

While FHSS lives on in some products, as I will explain later, it is not part of 802.11b. The only triplet anointed as part of the new standard is DSSS. This type of signalling uses a broadband carrier, generating a redundant bit pattern (called a chip) for every bit of data to be transmitted. While seemingly wasteful of bandwidth, DSSS copes well with weak signals: Data can often be extracted from a background of interference and noise without having to be retransmitted, making actual throughput superior.

Proponents of DSSS point to its superior range, plus its ability to reject multipath and other forms of interference. In fact, DSSS can reject noise from a microwave oven, for example, with relative ease, though it would still be swamped if deployed in the vicinity of a hospital's MRI scanner.

In any case, the 802.11b version of DSSS transmits data at a nominal 11Mbps (actual rates vary according to distance from another transmitter/receiver). It is downwardly compatible with 1Mbps and 2Mbps wireless networking products, provided they also use DSSS and are 802.11-compatible.

With few exceptions, 802.11b is a worldwide standard. It uses the 2.4GHz to 2.48GHz Instrumentation, Scientific and Medical (ISM) frequency band, dividing this into as many as 14 different channels. In the United States, 11 channels are available for use.

Vendors must tailor their hardware access points to use legal channels in each country they ship to. Wireless NICs, however, can often adapt themselves automatically to whatever channels are being employed locally. Therefore, it is possible to travel with an 802.11b client and make connections in any country.

It is difficult to plan a wireless network just by looking, or even by measuring distances. The antennas typically can, at the power levels permitted, transmit and receive for distances of about half a mile. This figure, however, only applies to outdoor, line-of-sight transmission.

Indoors, it is difficult to predict how a building's contour will affect propagation of radio waves. Range in an open plan building may be from 200 feet to 500 feet. In a closed-wall office environment, it may be as low as 100 feet. The metal found in an office building's floor can cut a signal by as much as 30 decibels (dB). Therefore, every floor in such a building will require one or more transmitters.

5.8.2.1.1 Basic Service Set The simplest type of wireless LAN is a peer-to-peer setup that might be used in a conference room or at a trade show. Here, all stations are kept within a circle with a radius of approximately 300 feet, and direct communication between stations is possible. To create this type of network, an administrator would install wireless NICs, setting their drivers to the ad hoc mode of operation, then selecting a radio channel for the workgroup.

In 802.11 lingo, this workgroup would be known as a Basic Service Set (BSS). A mechanism known as the Distributed Coordination Function (DCF), basically a "virtual carrier sense" function, provides best-effort delivery of data within a single, peer-to-peer BSS.

5.8.2.1.2 Access Point A more typical wireless network, however, is an "infrastructure" network - one that operates as an adjunct to a preexisting wired network. Here, Access Points (APs) are employed to act as a bridge (and usually a router), moving traffic between the wireless and wired networks.

A hardware AP is a self-contained unit, typically featuring one or more Ethernet ports, plus either a built-in radio or a PC Card slot. A software AP is a functional, more affordable equivalent, using an existing computer that has been equipped with both wired and wireless NICs to perform bridging and routing.

As well as providing a gateway between network types, an AP has several other functions. By broadcasting a beacon signal, the AP can temporarily silence ordinary terminals in order to provide point-to-point transmission of time-sensitive data, such as voice.

The primary functions of an AP, however, are authentication and association. The AP performs authentication to determine if a given wireless device is permitted to join the network, and can be based on MAC address, password, or some other parameter. Association is a handshaking relationship between the wireless device and the AP. It is designed to ensure that the client connects to only one AP at any given time.

5.8.2.1.3 Extended Service Set An Extended Service Set (ESS) is a logical collection of more than one BSS. Via an ESS, multiple APs can work together so that computers can roam from one to another while still staying in the same network.

Each 802.11 device associates with one AP initially, but a wireless network would be of limited use if stations were unable to roam. Fortunately, clients can switch from AP to AP in a way that is transparent to the user.

Logically, there are several ways roaming can take place, depending on the way APs have been set up. The simplest case is when different APs have the same ESSID and are on the same subnet of the same LAN.

Slightly more complexity results when different APs have the same ESSID but live on different subnets. Here, DHCP re-registration is required, unless a Mobile IP solution is being used. Multiple APs can also form different logical networks on a single LAN via the use of different ESSIDs.

Given the nature of radio-based communications, eavesdropping is always a possibility. Therefore, the 802.11 standard includes a shared-key encryption mechanism known as Wired Equivalent Privacy (WEP). When a client tries to connect to an AP, the AP sends a challenge value to the station. Upon receiving this, the client uses the shared key to encrypt the challenge and send it to the AP for verification.

5.8.2.1.4 Alternatives Of course, 802.11b is not the only entrant into the 2.4GHz wireless networking melee. A rival of sorts is the HomeRF Shared Wireless Access Protocol (SWAP) system, which has been designed for consumers. It uses FHSS transmission and eliminates the more complex parts of 802.11. An advantage here is that a single connection point can support both voice services via Time Division Multiple Access (TDMA) and data services via CSMA/CA.

Another contender, Bluetooth, uses the 2.4GHz band for localized connection between different devices. These might include a PC and a handheld device, a phone and a headset, or a notebook computer and a printer. While there are grounds for concern about interference between 802.11b, HomeRF, Bluetooth, and the many other devices using the same spectrum (such as baby monitors and garage door openers), some observers seem to believe all these can coexist.

Eventually, wireless LANs will migrate into the relatively wide-open spaces offered in the 5GHz band, where they will be able to exchange data at up to 54Mbps. Just as portable computers have always lagged behind their desktop cousins in terms of speed and affordability, wireless networks will always lag behind what copper and fiber can offer.

5.8.2.2 802.11 Media Access Control

The 802.11 MAC layer provides functionality to allow reliable data delivery for the upper layers over the wireless physical media. The data delivery itself is based on an asynchronous, best-effort, connectionless delivery of data. There is no guarantee that the frames will be delivered successfully.

The 802.11 MAC provides a controlled access method to the shared wireless media called Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA is similar to the collision detection access method deployed by 802.3 Ethernet LANs.

The third function of the 802.11 MAC is to protect the data being delivered by providing security and privacy services. Security is provided by the authentication services and by Wireless Equivalent Privacy (WEP), which is an encryption service for data delivered on the WLAN.

The fundamental access method of 802.11 is Carrier Sense Multiple Access with Collision Avoidance or CSMA/CA. CSMA/CA works by a "listen before talk scheme". This means that a station wishing to transmit must first sense the radio channel to determine if another station is transmitting. If the medium is not busy, the transmission may proceed.

The CSMA/CA protocol avoids collisions among stations sharing the medium by utilizing a random back-off time if the station's physical or logical sensing mechanism indicates a busy medium. The period of time immediately following a busy medium is the highest probability of collisions occurring, especially under high utilization. The CSMA/CA scheme implements a minimum time gap between frames from a given user. Once a frame has been sent from a given transmitting station, that station must wait until the time gap is up to try to transmit again. Once the time has passed, the station selects a random amount of time (the back-off interval) to wait before "listening" again to verify a clear channel on which to transmit. If the channel is still busy, another back-off interval is selected that is less than the first. This process is repeated until the waiting time approaches zero and the station is allowed to transmit. This type of multiple access ensures judicious channel sharing while avoiding collisions.

The optional request-to send and clear-to-send (RTS/CTS) function allows the access point to control use of the medium for stations activating RTS/CTS. With most radio NICs, users can set a maximum frame length threshold whereby the radio NIC will activate RTS/CTS. For example, a frame length of 1,000 bytes will trigger RTS/CTS for all frames larger than 1,000 bytes. The use of RTS/CTS alleviates hidden node problems, that is, where two or more radio NICs can't hear each other and they are associated with the same access point.

If the radio NIC activates RTS/CTS, it will first send a RTS frame to access point before sending a data frame. The access point will then respond with a CTS frame, indicating that the radio NIC can send the data frame. With the CTS frame, the access point will provide a value in the duration field of the frame header that holds off other stations from transmitting until after the radio NIC initiating the RTS can send its data frame. This avoids collisions between hidden nodes. The RTS/CTS handshake continues for each frame, as long as the frame size exceeds the threshold set in the corresponding radio NIC.

5.9 DSL

Digital Subscriber Line (DSL) technology is a modem technology that allows high bit rates across existing copper twisted-pair telephone lines. This allows high speed access between business/home and a network service provider's offices. From the provider's offices onward, the traffic can be carried over the service provider's high speed network (which may use technologies such as ATM). The DSL family of protocols (generically termed xDSL) covers a number of forms of DSL, including ADSL, SDSL, HDSL, RADSL, and VDSL. The primary advantage of xDSL is the provision of high-bandwidth directly to customer premises with minimal changes to existing telecommunications provider infrastructure.

A pair of wires, moderately twisted for the entire length between the telephone company's end office and the user premises (the common telephone set) form a loop, so it is referred to as the local loop. This loop provides a user with access to the global telecommunications infrastructure that is installed all over the world. The local loop has been historically designed to provide voice grade audio service. It is this existing local loop infrastructure that xDSL services are taking advantage of.

Many business and personal users are adopting ADSL, which offers high speed Internet connections, piggybacked on top of the existing telephone service and infrastructure.

5.9.1 Asymmetric Digital Subscriber Line (ADSL)

ADSL technology is asymmetric in that it offers more bandwidth downstream, from an service provider's central office to the customer site, than upstream from the subscriber to the central office. This asymmetry makes ADSL ideal for Internet surfing, and other client based based applications (such as downloading files from a company LAN). Many applications of this nature typically download more data than they upload.

Technically the asymmetry was introduced because of the nature of the last loop cabling. It is primarily due to crosstalk. The large bundle of wire at the service provider is heavily susceptible to crosstalk, particularly with regards to the signal that travels from the far end (the end user).

High bit rates, or in this case, higher frequencies suffer a greater amount of attenuation. The reason that the upstream speed in ADSL is generally much less than the downstream rate is due to this fact. When the high frequencies have attenuated at the service provider end, they are very susceptible to all the other signals in the wire bundle.

In the downstream direction, the high frequencies still attenuate, but at the customer end, they have a better chance of avoiding crosstalk since most subscribers will not have large bundles of cables running into their premises.

The use of error correction on ADSL connections makes it ideal of multimedia and real-time traffic, since damage to data can be fixed (to a degree) without requiring timeouts and retransmission. ADSL modems incorporate forward error correction that dramatically reduces errors caused by impulse noise. Error correction on a symbol-by-symbol basis also reduces errors caused by continuous noise coupled into a line.

ADSL transmits more than 6 Mbps to the client, while retaining an upload facility of up to 640 kbps. This is substantially more than can be achieved with analog modems, or even with other digital services such as ISDN.

An ADSL circuit connects an ADSL modem on each end of a twisted-pair telephone line, creating three information channels:

- a high-speed downstream channel
- a medium-speed duplex channel

Data Rate/[Mbps]	Wire Size/[mm]	Distance/[km]
2	0.5	5.5
2	0.4	4.6
6.1	0.5	3.7
6.1	0.4	2.7

Table 5.6: ADSL performance with distance.

- a basic telephone service channel.

The basic telephone service channel is split off from the digital modem by filters, thus guaranteeing uninterrupted basic telephone service, even if ADSL fails.

ADSL modems accommodate Asynchronous Transfer Mode (ATM) transport with variable rates and compensation for ATM overhead, as well as IP protocols.

Downstream data rates depend on a number of factors, including the length of the copper line, its wire gauge, presence of bridged taps, and cross-coupled interference. Line attenuation increases with line length and frequency and decreases as wire diameter increases. Typical distances for ADSL transmissions are shown in Table 5.6. These distances can cover up to 95% of the copper cables used by a telecommunications operator, supporting the widespread deployment of ADSL.

5.9.2 ADSL Technology

ADSL is a huge step in the data communication field, and it uses state of the art technology in more then one area. The main idea was to be able to work in larger bit rates with more reliability over the present copper lines technology. There are few basic requirements that ADSL had to answer:

- Full use of the copper line frequency spectrum (1.1MHz) .
- An advanced coding/decoding method.
- Ability to work simultaneously with POTS on the same copper line.

Copper lines have a frequency spectrum of 1.1MHz which can be used to data communication. The restrictions, however, are that the lower 4kHz must be retained for telephone service, and that the amplification isn't the same in all frequencies.

The technology being used is Discrete Multitone which divides the frequency range into 256 sub-frequencies (of 4kHz) from 64kHz to 1.1MHz Each sub-frequency is an independent channel and has it own stream of signals. The ADSL protocol defines a basic stream of data which is known to both endpoints in advanced and enables them to find the quality of transmission at each sub-frequency, and uses this information to split the data over the sub-frequencies. ADSL splits off a 4 kHz region for basic telephone service at the DC end of the band.

The Discrete Multitone technology is also very useful in the asymmetric mode where the sub-channels are divided to groups, one for the upstream data and the other for the downstream.

One of the most important technology breakthrough that helped the ADSL is the coding. Using a method called echo cancellation, the information on the line can be damaged during transmission and yet the decoder can still rebuild the information with very high reliability. Another useful method to increase the reliability of ADSL systems is Forward Error Correction (FEC).

As with most communication protocols ADSL uses a specific framing method. An ADSL modem organizes the aggregate data stream created by multiplexing downstream channels, duplex channels, and maintenance channels together into blocks, and attaches an error correction code to each block. The receiver then corrects errors that occur during transmission up to the limits implied by the code and the block length. The unit may, at the user's option, also create superblocks by interleaving data within subblocks; this allows the receiver to correct any combination of errors within a specific span of bits. This in turn allows for effective transmission of both data and video signals.

5.9.3 Very-High-Data-Rate Digital Subscriber Line (VDSL)

Networks with even higher speeds will be required in the future. Provision of bandwidth directly to the consumer's home and office requires that appropriate technology be available for the last hop between network service provider and the customer's premises. Future DSL technologies are being developed to provide for this.

Fiber all the way to the home (FTTH) is still prohibitively expensive in a marketplace soon to be driven by competition rather than cost. An attractive alternative is a combination of fiber cables feeding neighborhood optical network units (ONUs) and last-leg-premises connections by existing or new copper. This topology, which is often called fiber to the neighborhood (FTTN), encompasses fiber to the curb (FTTC) with short drops and fiber to the basement (FTTB), serving tall buildings with vertical drops.

VDSL is a new technology which is expected to serve in the FTTN role. VDSL provides high-speed data over short reaches (shorter than ADSL) of twisted-pair copper telephone lines, with a range of speeds depending on actual line length. Downstream rates of about 52 Mbps and upstream rates of about 2Mbps are expected over lines of about 300 m in length. It will also continue to operate over larger distances, but at lower data rates: typically downstream speeds around 13 Mbps for lengths of 1500 m.

As for ADSL, the data channels are separated in frequency from bands used for basic telephone service and Integrated Services Digital Network (ISDN). This allows these services to be retained, even when adding VDSL facilities.

Chapter 6

Network Protocols: Network Layer

6.1 Chapter Structure

6.1.1 Outcomes

1. Be able to construct, analyze and critically comment on network protocols and protocol design issues.
2. Appreciate the influence that the choice of protocol has on the performance of computer networks.
3. Select protocols that allow communication with others using a computer network.
4. Be able to classify the network protocols presented in this chapter in terms of the OSI layers.
5. Demonstrate the ability to access new information from the Internet particularly with respect to standards for Internet protocols.
6. Incorporate strategies used in these protocols to solve other communications and network related problems.
7. Appreciate the relevance of the various protocols, particularly with respect to their age.

6.1.2 Objectives

This chapter:

1. Describes various protocol stacks in common use, and the more significant protocols found within these stacks.
2. Describes how each of these protocols solves the communication issue(s) associated with its OSI layer.
3. Show how a protocol accomplishes its tasks in terms of the exchange of specific fields.
4. Show that Internet protocols have readily accessible standards documents available via the Internet.

6.2 Protocol stacks

Each network operating system manufacturer has implemented its own networking protocols to provide the required networking functions. These protocols operate as distinct programs or processes that the system uses to transport data between the network nodes. Each set of programs is commonly referred as a protocol stack (see Table 6.1). It is important to note that although the underlying functionality of each of these protocol stacks is similar, the implementation within each network system is unique.

OSI Model	Application	Presentation	Session	Transport	Network	Data Link	Physical
Banyan Vines	Vines Redirector	NetRPC Direct Socket		SPP & JPC	Vines IP ICP	ARP&RARP Vines Drivers	NIC
NT/Lan Manager	Server Message Block		NETBIOS Named Pipes	NETBEUI		NDIS	NIC
Novell Netware	Netware Core Protocols			SPX	IPX	ODI/NDIS	NIC
TCP/IP Unix	Network Applications	Socket Interface		TCP UDP	IP ICMP	ARP&RARP NDIS	NIC

Table 6.1: Layers of common protocols

A client application sends data down its protocol stack, passing through each of the protocols and interfaces. Information necessary to forward the application data to its destination is added by the programs operating at each level. At the receiving side, the data packets traverse a similar stack of protocols and programs, this time in reverse. Starting at the physical layer, the packet passed through each successive layer until it reaches the top of the stack at the relevant application process. At each layer, the information appended by the different protocols is examined so that the host can forward the packet to its final destination. For the host to accomplish this, both the client and the host need to run the same program at each level. If the server received a data packet that contained protocol information generated from a program not in its protocol stack, it would obviously not be able to understand the contained information.

Each subsequent layer, additional protocol information is appended to the original data packet. At the host side, the protocol information is stripped away layer by layer to finally leave the application data.

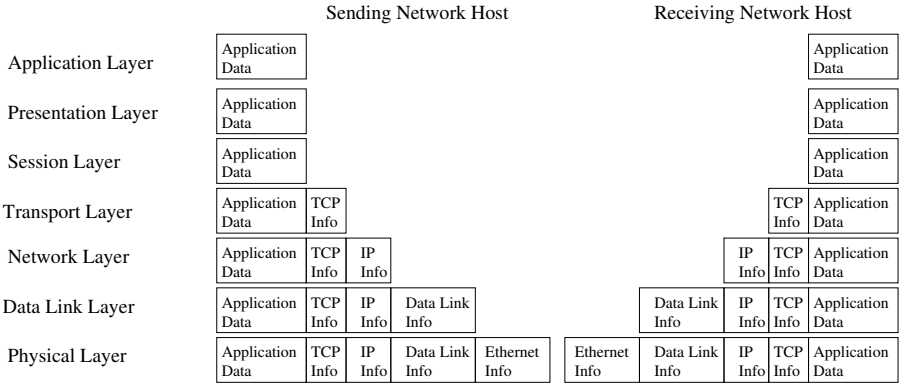


Figure 6.1: TCP/IP packet moving through the protocol layers

Figure 6.1 shows a more specific example of an application packet moving through a TCP/IP network.

The relationship between the various protocols in the TCP/IP suite of networked applications is illustrated in Table 6.2.

Session	Telnet	FTP	Gopher	SMTP	HTTP	DNS	SNMP	RIP	Ping
Transport	TCP	UDP							ICMP
Network	IP								
Data Link	Ethernet	Token Ring		FDDI	ISDN		ATM	SLIP	PPP

Table 6.2: TCP/IP related protocols

6.3 The Internet protocols

The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.

The actual technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). Much of the work is handled via mailing lists. The IETF holds meetings three times per year.

Each distinct version of an Internet standards-related specification is published as part of the "Request for Comments" (RFC) document series. This archival series is the official publication channel for Internet standards documents and other publications of the IESG, IAB, and Internet community. RFCs can be obtained from a number of Internet hosts using anonymous FTP, gopher, World Wide Web, and other Internet document-retrieval systems. In particular, they can all be accessed from the IETF web site at http://www.ietf.org .

Some RFCs document Internet Standards. These RFCs form the 'STD' sub-series of the RFC series. When a specification has been adopted as an Internet Standard, it is given the additional label "STDxxx", but it keeps its RFC number and its place in the RFC series.

Some of the better known Internet protocols are shown together with their RFC numbers in the list below:

RFC_768: User Datagram Protocol (UDP)

RFC 791: Internet Protocol (IP)

RFC 792: Internet Control Message Protocol (ICMP)

RFC_793: Transmission Control Protocol (TCP)

You should also keep an eye on where the protocols are going:

RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms

RFC_2002: IP Mobility Support

RFC_2018: TCP Selective Acknowledgment Options

RFC_2414: Increasing TCP's Initial Window

RFC_2460: Internet Protocol, Version 6 (IPv6) Specification

RFC_2481: A Proposal to add Explicit Congestion Notification (ECN) to IP

RFC_2581: TCP Congestion Control

RFC_2582: The NewReno Modification to TCP's Fast Recovery Algorithm

The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols. The IANA is chartered by the Internet Society (ISOC) to act as the clearinghouse to assign and coordinate the use of numerous Internet protocol parameters.

6.3.1 Common protocols

Computers attached to an Ethernet can send application data to one another using high-level protocol software, such as the TCP/IP protocol suite used on the worldwide Internet. The high-level protocol packets are carried between computers in the data field of the frames.

Common Protocols are:

IP: Internet Protocol. The lowest layer protocol defined in TCP/IP. This is the base layer on which all other protocols mentioned herein are built. IP is often referred to as TCP/IP as well.

UDP: User Datagram Protocol. This is a connectionless protocol built on top of IP. It does not provide any guarantees on the ordering or delivery of messages. This protocol is layered on top of IP.

TCP: Transmission Control Protocol. TCP is a connection oriented protocol that guarantees that messages are delivered in the order in which they were sent and that all messages are delivered. If a TCP connection cannot deliver a message it closes the connection and informs the entity that created it. This protocol is layered on top of IP.

ICMP: Internet Control Message Protocol. ICMP is used for diagnostics in the network. The Unix program, ping, uses ICMP messages to detect the status of other hosts in the net. ICMP messages can either be queries (in the case of ping) or error reports, such as when a network is unreachable.

PPP Point-to-Point Protocol - A protocol for creating a TCP/IP connection over both synchronous and asynchronous systems. PPP provides connections for host to network or between two routers, It also has a security mechanism. PPP is well known as a protocol for connections over regular telephone lines using modems on both ends. This protocol is widely used for connecting personal computers to the Internet.

SLIP Serial Line Internet Protocol - A point-to-point protocol to use over a serial connection, a predecessor of PPP. There is also an advanced version of this protocol known as CSLIP (compressed serial line Internet protocol) which reduce overhead on a SLIP connection by sending just a header information when possible, thus increasing packet throughput.

FTP File Transfer Protocol - FTP enables transferring of text and binary files over TCP connection. FTP allows to transfer files according to a strict mechanism of ownership and access restrictions. It is one of the most commonly used protocols over the Internet now days.

Telnet Telnet is a terminal emulation protocol, defined in RFC854, for use over a TCP connection. It enables users to login to remote hosts and use their resources from the local host.

SMTP Simple Mail Transfer Protocol - This protocol is dedicated for sending Email messages originated on a local host, over a TCP connection, to a remote server. SMTP defines a set of rules which allows two programs to send and receive mail over the network. The protocol defines the data structure that would be delivered with information regarding the sender, the recipient (or several recipients) and, of course, the mail's body.

HTTP Hyper Text Transport Protocol - A protocol used to transfer hypertext pages across the world wide web. SNMP Simple Network Management Protocol - A simple protocol that defines messages related to network management. Through the use of SNMP network devices such as routers can be configured by any host on the LAN.

ARP Address Resolution Protocol - In order to map an IP address into a hardware address the computer uses the ARP protocol which broadcast a request message that contains an IP address, to which the target computer replies with both the original IP address and the hardware address.

NNTP Network News Transport Protocol - A protocol used to carry USENET posting between News clients and USENET servers.

6.3.2 ARP - Address Resolution Protocol

High-level protocols have their own system of addresses, such as the 32-bit address used in the current version of IP. The high-level IP-based networking software in a given station is aware of its own 32-bit IP address and can read the 48-bit Ethernet address of its network interface, but it doesn't know what the Ethernet addresses of other stations on the network may be.

To make things work, there needs to be some way to discover the Ethernet addresses of other IP-based stations on the network. For several high-level protocols, including TCP/IP, this is done using yet another high-level protocol called the Address Resolution Protocol (ARP). As an example of how Ethernet and one family of high-level protocols interact, let's take a quick look at how the ARP protocol functions.

The operation of ARP is straightforward. Let's say an IP-based station (station "A") with IP address 192.0.2.1 wishes to send data over the Ethernet channel to another IP-based station (station "B") with IP address 192.0.2.2. Station "A" sends a packet to the broadcast address containing an ARP request. The ARP request basically says "Will the station on this Ethernet channel that has the IP address of 192.0.2.2 please tell me what the address of its Ethernet interface is?"

Since the ARP request is sent in a broadcast frame, every Ethernet interface on the network reads it in and hands the ARP request to the networking software running on the station. Only station "B" with IP address 192.0.2.2 will respond, by sending a packet containing the Ethernet address of station "B" back to the requesting station. Now station "A" has an Ethernet address to which it can send data destined for station "B," and the high-level protocol communication can proceed.

A given Ethernet system can carry several different kinds of high-level protocol data. For example, a single Ethernet can carry data between computers in the form of TCP/IP protocols as well as Novell or AppleTalk protocols. The Ethernet is simply a trucking system that carries packages of data between computers; it doesn't care what is inside the packages.

6.3.3 IP - Internet Protocol

The Internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The Internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.

The Internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an Internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols. The Internet protocol can capitalize on the services of its supporting networks to provide various types and qualities of service.

IP addressing is based on the concept of hosts and networks. A host is essentially anything on the network that is capable of receiving and transmitting IP packets on the network, such as a workstation, server or a router. The hosts are connected together by one or more networks. The IP address of any host consists of its network address plus its own host address on the network. IP addressing, unlike, say, IPX addressing, uses one address containing both network and host address.

An IP address is 32 bits wide, and is composed of two parts: the network number, and the host number. By convention, it is expressed as four decimal numbers separated by periods, such as "200.1.2.3" representing the decimal value of each of the four bytes. Valid addresses thus range from 0.0.0.0 to 255.255.255.255, a total of about 4.3 billion addresses. The first few bits of the address indicate the Class that the address belongs to:

Class	Prefix	Network Number	Host Number
A	0	Bits 1-7	Bits 8-31
B	10	Bits 2-15	Bits 16-31
C	110	Bits 3-23	Bits 24-31
D	1110	N/A	
E	1111	N/A	

Class D addresses are multicast, and Class E are reserved. Any address starting with 127 is a loopback address and should never be used for addressing outside the host. A host number of all binary 1's indicates a directed broadcast over the specific network. For example, 200.1.2.255 would indicate a broadcast over the 200.1.2 network. If the host number is 0, it indicates "this host". If the network number is 0, it indicates "this network".

The format of an IP header is shown in Table 6.3.

- Version: 4 bits - indicates the format of the Internet header. This document describes version 4.

Bits 0-7		Bits 8-15	Bits 16-23	Bits 24-31
Version	IHL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				Padding

Table 6.3: IP Header

- IHL: 4 bits - Internet Header Length is the length of the Internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.
- Type of Service: 8 bits - The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic (generally by accepting only traffic above a certain precedence at time of high load).
- Total Length: 16 bits - Total Length is the length of the datagram, measured in octets, including Internet header and data. This field allows the length of a datagram to be up to 65,535 octets.
- Identification: 16 bits - An identifying value assigned by the sender to aid in assembling the fragments of a datagram.
- Flags: 3 bits - Various Control Flags.
- Fragment Offset: 13 bits - This field indicates where in the datagram this fragment belongs.
- Time to Live: 8 bits - This field indicates the maximum time the datagram is allowed to remain in the Internet system. If this field contains the value zero, then the datagram must be destroyed. This field is modified in Internet header processing. The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.
- Protocol: 8 bits - This field indicates the next level protocol used in the data portion of the Internet datagram.
- Header Checksum: 16 bits - A checksum on the header only. Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the Internet header is processed. The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.
- Source Address: 32 bits
- Destination Address: 32 bits
- Options: variable - The options may appear or not in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation.
- Padding: variable - The Internet header padding is used to ensure that the Internet header ends on a 32 bit boundary. The padding is zero.

6.3.4 UDP - User Datagram Protocol:

UDP gives application programs direct access to a datagram delivery service, like the delivery service that IP provides. This allows applications to exchange messages over the network with a minimum of protocol overhead. UDP is an unreliable (it doesn't care about the quality if deliveries it make), connectionless (doesn't establish a connection on behalf of user applications) datagram protocol. Within your computer, UDP will deliver data correctly. UDP is used as a data transport service when the amount of data being transmitted is small, the overhead of creating connections and ensuring reliable delivery may be greater than the work of retransmitting the entire data set. Broadcast-oriented services use UDP, as do those in which repeated, out of sequence, or missed requests have no harmful side effects. Since no state is maintained for UDP transmission, it is ideal for repeated, short operations such as the Remote Procedure Call protocol. UDP packets can arrive in any order. If there is a network bottleneck that drops packets, UDP packets may not arrive at all. It's up to the application built on UDP to determine that a packet was lost, and to re-send it if necessary.

NFS and NIS are build on top of UDP because of its speed and statelessness. While the performance advantages of a fast protocol are obvious, the stateless nature of UDP is equally important. Without state information in either the client or server, crash recovery is greatly simplified. The structure of a UDP

Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31
Source Port		Destination Port	
Length		Checksum	

Table 6.4: UDP Datagram Header

packet header is shown in Table 6.4.

- Source Port (16 bits): This field is optional and specifies the port number of the application that is originating the user data.
- Destination Port (16 bits): This is the port number pertaining to the destination application.
- Length (16 bits): This field describes the total length of the UDP datagram, including both data and header information.
- UDP checksum (16 bits): Integrity checking is optional under UDP. If turned on, this field is used by both ends of the communication channel for data integrity checks.

6.3.5 TCP - Transmission Control Protocol

TCP is a fully reliable, connection-oriented, acknowledged, byte stream protocol that provide reliable data delivery across the network and in the proper sequence. TCP supports data fragmentation and reassembly. It also support multiplexing/demultiplexing using source and destination port numbers in much the same way they are used by UDP.

TCP provides reliability with a mechanism called Positive Acknowledgment with Retransmission (PAR). Simply stated, a system using PAR sends the data again, unless it hears from the remote system that the data arrived OK. The unit of data exchanged between co-operating TCP modules is called a segment.

The structure of a TCP packet header is shown in Table 6.5.

- Source port (16 bits): Specifies the port on the sending TCP module.
- Destination port (16 bits): Specifies the port on the receiving TCP module.
- Sequence number (32 bits): Specifies the sequence position of the first data octet in the segment. When the segment opens a connection, the sequence number is the Initial Sequence Number (ISN) and the first octet in the data field is at sequence ISN+1

Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31
Source Port		Destination Port	
Sequence Number			
Acknowledgment Number			
Offset	Reserved	Control	Window
Checksum		Urgent Pointer	
Options			Padding

Table 6.5: TCP Packet Header

- Acknowledgment number (32 bits): Specifies the next sequence number that is expected by the sender of the segment. TCP indicates that this field is active by setting the ACK bit, which is always set after a connection is established.
- Data offset (4 bits): Specifies the number of 32-bit words in the TCP header.
- Control bits (6 bits): The six control bits are as follow:
  - URG: When set, the Urgent Pointer field is significant
  - ACK : When set, the acknowledgment Number field is significant
  - PSH : Initiates a push function
  - RST : Forces a reset of the connection
  - SYN : Synchronizes sequencing counters for the connection. This bit is set when a segment request opening of a connection.
  - FIN : No more data. Closes the connection
- Window (16 bits): Specifies the number of octets, starting with the octet specified in the acknowledgment number field, which the sender of the segment can currently accept.
- Checksum (16 bits): An error control checksum that covers the header and data fields.
- Urgent Pointer (16 bits): Identifies the sequence number of the octet following urgent data. The urgent pointer is a positive offset from the sequence number of the segment.
- Options (variable): Options are available for a variety of functions.
- Padding (variable): 0-value octets are appended to the header to ensure that the header ends on a 32-bit word boundary.

TCP is connection-oriented. It establishes a logical end-to-end connection between the two communication hosts. Control information, called a handshake, is exchanged between the two endpoints to establish a dialog before data is transmitted. TCP indicates the control function of a segment by setting the appropriate bit in the flags field of the segment header.

The type of handshake used by TCP is called a three-way handshake because three segments are exchanged. Host A sends a SYN to host B, host B responds with a SYN,ACK and host A acknowledges that with an ACK and begins data transfer.

TCP employs the positive acknowledgment with retransmission technique for the purpose of archiving reliability in service. When TCP send a data segment, it requires an acknowledgment from the receiving end. The acknowledgment is used to update the connection state table. An acknowledgment can be positive or negative. An positive acknowledgment implies that the receiving host recovered the data and that it passed the integrity check. A negative acknowledgment implies that the failed data segment needs to be retransmitted. It can be caused by failures such as data corruption or loss.

TCP detects when a packet is lost on the network and fails to reach its ultimate destination. When a host sends data, it starts a count down timer. If the timer expires without receiving an acknowledgment, this

host assumes that the data segment was lost. Consequently, this host retransmits a duplicate of the failing segment. TCP keep a copy of all transmitted data with outstanding positive acknowledgment. Only after receiving the positive acknowledgment is this copy discarded to make room for other data in its buffer.

6.3.6 ICMP - Internet Control Message Protocol

Occasionally a gateway or destination host will communicate with a source host, for example, to report an error in datagram processing. For such purposes this protocol, the Internet Control Message Protocol (ICMP), is used. ICMP, uses the basic support of IP as if it were a higher level protocol, however, ICMP is actually an integral part of IP, and must be implemented by every IP module.

ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route.

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

The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages. Also ICMP messages are only sent about errors in handling fragment zero of fragmented datagrams. (Fragment zero has the fragment offset equal zero).

ICMP messages may fall into the following categories:

- Destination Unreachable Message: If, according to the information in the gateway's routing tables, the network specified in the Internet destination field of a datagram is unreachable, e.g., the distance to the network is infinity, the gateway may send a destination unreachable message to the Internet source host of the datagram. In addition, in some networks, the gateway may be able to determine if the Internet destination host is unreachable. Gateways in these networks may send destination unreachable messages to the source host when the destination host is unreachable.

- Time Exceeded Message: If the gateway processing a datagram finds the time to live field is zero it must discard the datagram. The gateway may also notify the source host via the time exceeded message.

- Parameter Problem Message: If the gateway or host processing a datagram finds a problem with the header parameters such that it cannot complete processing the datagram it must discard the datagram. One potential source of such a problem is with incorrect arguments in an option.

- Source Quench Message: A gateway may discard Internet datagrams if it does not have the buffer space needed to queue the datagrams for output to the next network on the route to the destination network. If a gateway discards a datagram, it may send a source quench message to the Internet source host of the datagram. A destination host may also send a source quench message if datagrams arrive too fast to be processed. The source quench message is a request to the host to cut back the rate at which it is sending traffic to the Internet destination.

- Redirect Message: The gateway sends a redirect message to a host in the following situation. A gateway, G1, receives an Internet datagram from a host on a network to which the gateway is attached. The gateway, G1, checks its routing table and obtains the address of the next gateway, G2, on the route to the datagram's Internet destination network, X. If G2 and the host identified by the Internet source address of the datagram are on the same network, a redirect message is sent to the host. The redirect message advises the host to send its traffic for network X directly to gateway G2 as this is a shorter path to the destination.

- Echo or Echo Reply Message: The data received in the echo message must be returned in the echo reply message. The identifier and sequence number may be used by the echo sender to aid in matching the replies with the echo requests.

6.4 Proprietary protocols

6.4.1 IPX - Internetwork Packet Exchange

IPX is a networking protocol used by the Novell Netware operating systems. It acts as the datagram protocol for Novell, just as IP functions in that capacity for the Internet. Additional higher level protocols such as SPX (Sequenced Packet Exchange) and NCP are used to provide reliable connection oriented services (similar to TCP for the Internet).

An IPX address consists of a 4-byte Network Number, a 6-byte Node Number, and a 2-byte Socket Number. The node number is usually the hardware address of the interface card, and must be unique inside the particular IPX network. The network number must be the same for all nodes on a particular physical network segment. Socket numbers correspond to the particular service being accessed.

6.4.2 SMB - Server Message Block

SMB is a message format used by DOS and Windows to share files, directories and devices. SMB-based networks include Lan Manager, Windows for Workgroups, Windows NT, and Lan Server. There are also a number of products that use SMB to enable file sharing among different operating system platforms. A product called Samba, for example, enables UNIX and Windows machines to share directories and files.

Chapter 7

Networking Operations

This chapter deals with the mechanisms required to run and maintain a network.

7.1 Chapter Structure

7.1.1 Outcomes

1. Demonstrate the ability to access new information from appropriate sources, particularly the Internet.
2. Appreciate that there is a need for life-long learning to remain current in the field of computer networking.

7.1.2 Objectives

This chapter:

1. Provides detail on the routing and bridging processes referred to by other sections of the course.
2. Shows some of the material which needs to be studied in addition to that covered within the course.
3. Represents a starting point for exploring recent developments in the area of routing algorithms and standards.

7.2 Routing

Routing addresses the problem of getting packets from one end point to another. The mechanism used should be:

- Robustness: The world changes, software changes, use changes, topology and hardware change, things go wrong in lots of different ways. How well does the routing algorithm handle all this?
- Stability: Does the algorithm find a routing table quickly (convergence)? How does it adapt to abrupt changes in topology or state of the routers? Is it possible to have oscillations?
- Fairness & Optimality.

Algorithms may be static, i.e. the routing decisions are made ahead of time, with information about the network topology and capacity, then loaded into the routers, or dynamically, where the routers make decisions based on information they gather, and the routes change over time, adaptively.

Links between routers have a cost associated with them. In general it could be a function of distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, router processing speed, etc.

Two basic approaches can be used for routing within an network :

1. Centralized routing where the routing information associated with each gateway is downloaded from a central site using network and special management messages. The network management system endeavors to maintain the contents of the routing database and to keep it up to date when hosts and networks are added or withdrawn. In general for all but the smallest internets, this is the only viable solution as long as each network has its own management software.

2. Distributed routing is where all the hosts and gateways cooperate in a distributed way to ensure that the routing information held by each system is always up to date. Each system hold relevant routing information in routing table. The Internet uses such a scheme.

7.2.1 Shortest Path Algorithm (Dijkstra's algorithm)

The shortest path algorithm just finds the least expensive path through the network, based on the cost function.

Dijkstra's algorithm is an example. You can start from source/destination (doesn't matter in a unidirectional graph).

1. Set probe node to starting node.

2. Probe neighboring nodes and tentatively label them with (probe node, cumulative distance from start).

3. Search all tentatively labeled nodes (and not just the nodes labeled from the current probe) for the minimum label, make this minimum node's label permanent, and make it the new probe node.

4. If the probe node is the destination/source, stop, else goto 2.

The distance part of the node labels is cumulative distance from the starting node, not simply distance from the last probe node.

The key to discovering that you've gone down a bad (greater distance) path is that you examine all nodes with temporary labels in step 3. This means that you switch the probe node to another, shorter path if the you run into an high cost link.

If you label each node with it's predecessor on the path, and the distance to that node, then you can easily find the route you desire (albeit backwards) by starting at the destination and following the trail of predecessors backwards. You'll also know the distance from source to destination from the label on the destination.

7.2.2 Flooding

Every incoming packet is sent out on every other link by every router.

1. Origin node sends packets to its neighbors

2. Neighbors relay to their neighbors following the rules:

   (a) A node does not relay packet to a node from which the packet was obtained (Ensures no cycles)

   (b) A node transmits the packet only once, achieved by adding packet sequence numbers.

3. If L is the number of bidirectional links in the network, then $L \leq \mathit{Packet\,Transmissions} \leq 2L$

Super simple to implement, but generates lots of redundant packets. Interesting to note that all routes are discovered, including the optimal one, so this is robust and high performance (best path is found without being known ahead of time). Good when topology changes frequently.

Some means of controlling the expansion of packets is needed. Could try to ensure that each router only floods any given packet once.

Could try to be a little more selective about what is forwarded and where.

7.2.3 Flow-based

Similar in spirit to minimum distance, but takes traffic flow into consideration.

The key here is to be able to characterize the nature of the traffic flows over time. You might be able to do this if you know a lot about how the network is used (traffic arrival rates and packet lengths). From the known average amount of traffic and the average length of a packet you can compute the mean packet delays using queuing theory. Flow-based routing then seeks to find a routing table to minimize the average packet delay through the subnet.

7.2.4 Deflection Routing

In deflection routing, nodes try to get rid of the packets as soon as possible (to avoid packet loss) to any idle outgoing link. A packet can take any free outgoing link.

- Good for networks where nodes have shortage of buffer space.
- Different packets may take different routes.
- Some packets may travel a long path.
- Packets may be received out-of-order.
- Precautions must be taken to avoid cycles

7.2.5 Distance Vector

Also known as Belman-Ford or Ford-Fulkerson. Used in the original ARPANET, and in the Internet as RIP.

The heart of this algorithm is the routing table maintained by each host. The table has an entry for every other router in the subnet, with two pieces of information: the link to take to get to the router, and the estimated distance from the router. For a router A with two outgoing links L_1, L_2, and a total of four routers in the network, the routing table might look like this:

Router	Distance	Link
B	5	L_1
C	7	L_1
D	2	L_2

Neighboring nodes in the subnet exchange their tables periodically to update each other on the state of the subnet (which makes this a dynamic algorithm). If a neighbor claims to have a path to a node which is shorter than your path, you start using that neighbor as the route to that node. Notice that you don't actually know the route the neighbor thinks is shorter - you trust his estimate and start sending frames that way.

When a neighbor sends you its routing table you examine it as follows and update your own routing table.

```
for( i varied across all routers in the table )
  if( your distance to the neighbor +
      neighbors distance to router i <
        your previous estimate to router i )
    {
      your distance to router i = your distance to the neighbor +
        neighbors distance to router i
      link to router i is set to link to the neighbor with the
        short distance to i
    }
```

You can think of this as forming an approximation of the global state of the subnet from local information only (exchange with neighbors). Unfortunately it has problems (it's only an approximation, after all). Good news (a link comes up, a new router is available, a router or link are made faster) propagate very quickly through the whole subnet (in the worst case it takes a number of exchanges equal to the longest path for everyone to know the good news).

Bad news is not spread reliably. Neighbors only slowly increase their path length to a dead node, and the condition of being dead (infinite distance) is reached by counting to infinity one at a time. Various means of fixing this have been tried, but none are foolproof.

7.2.6 Link State

Widely used today, replaced Distance Vector in the ARPANET. Link State improves the convergence of Distance Vector by having everybody share their idea of the state of the net with everybody else (more information is available to nodes, so better routing tables can be constructed).

The basic outline is

1. discover your neighbors
2. measure delay to your neighbors
3. bundle all the information about your neighbors together
4. send this information to all other routers in the subnet
5. compute the shortest path to every router with the information you receive

7.2.6.1 Neighbor discovery

Send an HELLO packet out. Receiving routers respond with their addresses, which must be globally unique.

7.2.6.2 Measure delay

Time the round-trip for an ECHO packet, divide by two. Question arises: do you include time spent waiting in the router (i.e. load factor of the router) when measuring round-trip ECHO packet time or not?

7.2.6.3 Bundle your info

Put information for all your neighbors together, along with your own id, a sequence number and an age.

7.2.6.4 Distribute your info

Ideally, every router would get every other routers data simultaneously. This can't happen, so in effect you have different parts of the subnet with different ideas of the topology of the net at the same time. Changes ripple through the system, but routers that are widely spread can be using very different routing tables at the same time. This could result in loops, unreachable hosts, other types of problems.

A flooding algorithm is used to get the data out as soon as possible. The sequence number is used to control the flooding. Each router keeps track of the routing data packets it has seen by router/sequence no. pair. If the pair is new, then it is forwarded on all lines but the one it arrived on, if it has been seen before it is not forwarded.

The age of a data packet is used to prevent corruption of the sequence number from causing valid data to be ignored. The age field is decremented once per second by the routers which forward the packet. When it hits zero it is discarded.

7.2.6.5 Compute shortest path tree

Using an algorithm like Dijkstra's, and with a complete set of information packets from other routers, every router can locally compute a shortest path to every other router. The memory to store the data is proportional to $k \times n$, for n routers each with k neighbors. Time to compute can also be large.

Bad data (from routers in error, e.g.) will corrupt the computation.

7.2.7 Hierarchical

When your subnet is large then the routing tables become unwieldy. Too much memory to store them, too much time to search them, too much time to compute them. When something is too large, people form a hierarchy to deal with it.

The idea is to replace N different routing table entries to N different individual routers with a single entry for a cluster of N routers. You can apply many different levels of hierarchy.

The price you pay is that you don't have the optimal route for each router anymore (makes sense, since you lump all routers in a single cluster together with one optimal path).

7.2.8 Broadcast

Unicast routing is most general, but broadcast routing is a good match to many applications (e.g. distributing routing data, sending common information like weather reports).

One way of doing this is by sending a packet which has the desired data along with a list of hosts to be visited. At each router the list is examined. For each outgoing line which is the best path to one of the destinations in the list, the packet is duplicated and the destinations which are best sent down this line are selected from the list for the new packet. Eventually the packets only contain a single destination, in which case the algorithm has done the distribution.

7.2.9 Cut Through Routing

A node can transmit any portion of the packet without waiting to receive the entire packet. In this way pieces of the same packet may be simultaneously traveling on different links, while others are stored at different nodes. The link is reserved by the head end of the packet, and relinquished after the last bit is transmitted. This allows rapid transmission, with delay = Transmission on the slowest link + Propagation delay. Error detection and correction has to be done on an end-to-end basis (as opposed to link-by-link basis).

7.3 Bridging

If a network contains a loop, for example where two parallel bridges connect two LANs, then packet forwarding by one bridge can cause problems with the learning algorithm in the other. A loop occurs when there are alternate routes between hosts. If there is a loop in an extended network, bridges may forward traffic indefinitely, which can result in increased traffic and degradation in network performance.

You can avoid this problem by implementing the spanning tree algorithm, which produces a logical tree topology out of any arrangement of bridges. The result is that a single path exists between any two end stations on an extended network. The spanning tree algorithm also provides a high degree of fault tolerance. It allows the network to automatically reconfigure the spanning tree topology if there is a bridge or data-path failure.

7.3.1 Spanning Tree Algorithm

The spanning tree algorithm ensures the existence of a loop-free topology in networks that contain parallel bridges. In a physical network, it may be constructed as follows:

1. Tree rooted at A (say)

2. A sends packets to its neighbors
3. Neighbors send packets to their neighbors
4. All nodes mark the transmitter of the first packet they receive as their parent
5. Nodes relay packets to their neighbors only once

Other algorithms exist which allow construction using other costs.

7.3.1.1 The greedy algorithm

The first method for creating spanning trees is called the greedy algorithm. It works as follows:

1. Take the input, and sort the edges in order of increasing cost (breaking ties arbitrarily).
2. Now process the edges in this order.
3. Let T denote the set of edges that you have included in the minimum spanning tree thus far. Initially, set $T = \emptyset$. Let e denote the next cheapest edge.
4. Try adding edge e to the solution T that you have constructed thus far. If there is a cycle made up of edges only from $T \cup \{e\}$, then e is discarded; otherwise, we include e in T. We continue doing this until all of the edges have been processed.
5. The set T at the end specifies a minimum spanning tree.

This algorithm is also called Kruskal's algorithm.

7.3.1.2 Prim's algorithm

Here is another algorithm, which more closely resembles Dijkstra's algorithm. The idea is to "grow" the spanning tree at an arbitrarily chosen source node.

1. Pick any node to be the source s.
2. In general, we will maintain a set of nodes S that are already connected to s; that is, for each $v \in S$, there exists a path from s to v only using edges currently in T. We initially set $T = \emptyset$, and let $S = \{s\}$.
3. In each iteration, we try to find the cheapest edge e that connects up a new node; that is, we search for the cheapest edge that has one of its endpoints in S, and the other endpoint not in S. Suppose this edge is $[i, j]$, where $i \in S$ and $j \notin S$. Then we set $T = T \cup \{e\}$ and $S = S \cup \{j\}$.
4. We continue these iterations until S is equal to the entire set of nodes N.

7.3.1.3 Spanning tree solution to parallel bridges

The spanning tree algorithm requires five values to derive the spanning tree topology. The first, a multicast address specifying all bridges on the extended network, is media-dependent and is automatically determined by the software. You assign the remaining four values, which are

1. Network-unique identifier for each bridge on the extended network
2. Unique identifier for each bridge/LAN interface (called a port)
3. Priority specifying the relative priority of each port
4. Cost for each port

After you assign these values, bridges multicast and process the formatted frames (called Bridge Protocol Data Units, or BPDUs) to derive a single loop-free topology throughout the extended network. The bridges exchange BPDU frames quickly, minimizing the time that service is unavailable between hosts.

In constructing a loop-free topology, the bridges within the extended network follow these steps:

1. Elect a root bridge: The bridge with the lowest priority value becomes the root bridge and serves as the root of the loop-free topology. If priority values are equal, the bridge with the lowest bridge MAC address becomes the root bridge.

2. Determine path costs: The path cost is the cost of the path to the root bridge offered by each bridge port.

3. Select a root port and elect a designated bridge on each LAN: Each bridge designates the port that offers the lowest-cost path to the root bridge as the root port. In the event of equal path costs, the bridge examines the paths' interfaces to the root bridge. The port (interface) of the path with the lowest interface priority to the root bridge becomes the root port.

   If the paths' interfaces to the root bridge are also equal, then the root port is the port on the bridge with the lowest priority value.

   The spanning tree algorithm selects a bridge on each LAN as the designated bridge. The root port of this bridge has the lowest-cost path to the root bridge. All bridges turn off (set to blocking state) all of the lines except for the single line that is the shortest-cost path to the root and any line attached to the LANs for which the bridge serves as a designated bridge.

4. Elect a designated port: The spanning tree algorithm selects the port that connects the designated bridge to the LAN as the designated port. If there is more than one such port, the spanning tree algorithm selects the port with the lowest priority as the designated port. This port, which carries all extended network traffic to and from the LAN, is in the forwarding state.

Thus, the spanning tree algorithm removes all redundant ports (ports providing parallel connections) from service (places in the blocking state). If there is a topological change or a bridge or data-path failure, the algorithm derives a new spanning tree that may move some ports from the blocking to the forwarding state.

It is very important to configure the spanning tree parameters correctly. Consider the typical flow of traffic so that the logical topology that results from the spanning tree algorithm is appropriate for the network.

Chapter 8

Network Design

8.1 Chapter Structure

8.1.1 Outcomes

1. Demonstrate knowledge and understanding of the information, concepts and principles applicable to computer networking.
2. Be able to construct, analyze and critically comment on network design issues.
3. Assess whether a given computer network design satisfies specific criteria.
4. Ability to communicate computer network designs in appropriate written and diagrammatic form.
5. Appreciate that there is a need for life-long learning to remain current in the field of computer networking.

8.1.2 Objectives

This chapter:

1. Describes issues that are relevant when designing a computer network.
2. Discusses alternative scenarios and discusses the impact on each network design issue.
3. Presents design case studies and describes the degree to which they would meet various network design criteria.
4. Show appropriate diagrammatic layouts for presenting network designs.
5. Emphasizes the considerations involved in using cutting edge network technology.

8.2 Introduction

At the very least, the architecture is the foundation for your network design. Wise network designers realize the architecture maps your computing enterprise: In a distributed-computing environment, the network is the computer. Accordingly, the architecture should capture your corporate computing philosophies, strategies, and objectives. It must define where your network is now, where you want it to go, and how you plan to get there. It must also describe critical success factors, design objectives, and a logical network topology, as well as your strategies for network management and security.

8.3 Design Principles

There are two guidelines that often outweigh all others when developing a network: Keep it as simple and straightforward as possible; set and adhere to standards and disciplined practices. This is where network manageability begins.

8.3.1 Analyzing requirements

Never forget (as corporate IT sometimes does) that the successful enterprise- computing system serves the processing and information needs of the entire organization. Because the network architecture maps the distributed-computing enterprise, it should reflect the current and future structures and activities in every part of the organization. We define six key areas for analysis to help you identify the structure and computing activities of your enterprise and develop a set of network architecture requirements: corporate structure, desktop configurations, enterprise applications and services, workgroups, existing networks, and data centers and facilities.

8.3.1.1 Corporate Structure

What are the organizational units and how are they related to one another? How many users are in each unit and where are they located? Much of this information is available in a simple organizational chart. Knowing the organizational structure helps you define the size and extent of the network. It also helps you define the general layout of the network and potential technologies. If the organization is small and limited to one floor of a building, you might want to consider a simple Ethernet or Token Ring network. But if there are many departments spread across many floors, you might want to consider a larger network based on a corporate backbone that connects smaller networks in different areas. If there are departments spread over a wide geographical area, you will want to consider different WAN (wide area network) technologies to extend the backbone network to remote locations. Knowing your organizational structure also helps you define potential information flows and workgroups.

8.3.1.2 Desktop Configurations

Although "dumb" terminals are the mainstay of legacy systems, distributed-computing environments typically place much of the computing power right on users' desktops, where it belongs, to best serve productivity needs. But these desktop systems tend to be a variety of personal to workstation-class computers from a variety of hardware vendors running disparate operating systems and applications. Bringing this heterogeneous collection of desktops into the enterprise is no cakewalk (the user is willing, but the platform is often weak). From the networking standpoint, you need to identify the various hardware and software configurations that currently exist: those you actually can and intend to connect to the network (Commodore-64s make great doorstops) and the network adapters, software protocols, and network operating systems needed for the connection.

8.3.1.3 Enterprise applications and services

You need to identify existing applications and services as well as future requirements. These include databases and related client/server applications, file and print services, e-mail, and groupware. It is important to define the scope of applications and services. Are there enterprise applications, such as a large personnel database, accessed by users in many departments? Are there departmental applications limited to the users in a single department? And what about personal productivity applications on the desktops, such as word processors and spreadsheets? Will they be stored centrally on servers or distributed across the desktops? This information is important because it will affect the layout of the network. For example, you may want to place enterprise applications on the backbone and departmental applications on smaller networks in individual departments. Also, review user requirements for special applications and user plans for future applications development. These may influence the technologies you need to evaluate. For example, transaction processing, multimedia, and client/server applications will likely require faster network technologies.

8.3.1.4 Workgroups

The meteoric success of so-called "workgroup" software, such as Lotus Notes, highlights the fact that inter- and intradepartmental working collaborations are not only encouraged but de rigueur today. These small-to-medium-size groups share resources and information unique to their group and project. Although some workgroups, such as a sales team, can be permanently established, they tend to be transient, comprising consultants who leave the company and regular employees who are reassigned once the project is completed. Applications development is a good example of transient collaborations among design and programming teams.

Workgroups have a major impact on your network because their computing needs regularly cross the common boundaries of your organization's political and physical structures. To manage network traffic and improve performance, we recommend a networking topology that enables flexible workgroups. Hence, you need to discover the various workgroups, what unique applications their users require, and how they access the software across departmental or even worldwide servers. You also need to identify any special characteristics of the data traffic over the various segments of the network between individuals in a workgroup and, in particular, desired response times for that traffic.

8.3.1.5 Existing networks

If only we could start with a clean slate! The reality is, like the wild revolution that brought the conglomeration of PCs through the back door, departments and workgroups in most organizations have already taken the step and created a crazy quilt of networks throughout the enterprise. Like the collection of heterogeneous desktops, if you intend to iron things out into a cohesive networking architecture, first you've got to identify and classify those existing systems. Then you must decide which are worth connecting and which are doorstops. Are they Ethernet-based, Token Rings, or something else? What type of cabling is in place? What additional cabling will be required? What network protocols are being used? We find that mainframes typically use SNA; VMS systems usually use DECnet over Ethernet; Unix systems use TCP/IP; and Netware systems use IPX. Look, too, beyond LANs (local area networks) and identify the WANs. What types of WANs are in place, and what other options are available?

8.3.1.6 Data centers and facilities

Though we retain many of the concepts and disciplines of the legacy mainframe data center, we physically disperse the facilities in our distributed environments. In fact, just as the network enables a distributed environment, so too is the network infrastructure essential to the data center. The important issues are where to locate the facilities and how to build a WAN to support them. Are there sufficient conduits for cabling? Is there adequate power and air conditioning? And what about the wiring closets? Are they secure? Do they have sufficient power and ventilation? And what about racks and patch panels?

As you can see, there are a variety of issues to address in the network architecture for the New Enterprise. The issues raised here provide a good point of departure. As we proceed with refining the network architecture, there will be more detailed issues to address. After a series of iterations, we will eventually have a comprehensive design.

8.3.2 Design for success

After all the effort you and your colleagues will expend analyzing needs and developing your enterprise network, of course you want your network to work. But what makes a successful distributed network, and how will you know when you achieve it? Do you take a poll? Believe it or not, yes! Positive user opinion is a key criterion for success. That's why we involve users in the design and implementation process from the beginning. You probably won't be able to buy or steal success at the polls; instead, you'll have to earn approval the old-fashioned way by building for success.

From the user's perspective, the criteria for a successful network are functionality (Is the network useful and easy to use?), performance (Is it fast and responsive?), and reliability (Does it do what's expected and when it's needed?). Does that all sound familiar? (Hint: We're RASing you – reliability, availability, and serviceability. The fundamental disciplines for production-quality computing.)

8.3.2.1 Functionality, performance, reliability

Before you bend your designers and the network out of shape, remember: The simpler, the better. That's our number one design objective and why we use a structured architecture. It simplifies the network, thereby enhancing reliability, and supports new requirements and technologies without major modifications.

So, what exactly is a structured architecture? By structured we mean the network has some underlying order that is easy to understand and visualize. The network consists of many components. These components can be connected many ways, which add to the network's complexity. We can simplify the network by introducing order or structure. Simplifying the network makes it more manageable. For example, instead of connecting many networks in a disorganized web, we use a backbone network. The backbone provides a common way to interconnect the many smaller networks, which is easier to visualize and manage than a web of networks with many unorganized interconnections. A structure doesn't just happen; structures are planned, and they must be planned from the beginning.

Other critical design features that get users to pull the "yea" lever in the voting booth are connectivity, interoperability, and transparency. Connecting the best computing resources over the network, regardless of the origins, brings the entire enterprise to users' desktops. Do it in a way that hides the complexities and technical complications, and the network becomes comprehensible and easy to use. Hence, a network that is transparent to users and connects the various heterogeneous platforms throughout the enterprise enhances the critical-success criterion of functionality.

Once elected to office (by a landslide, of course), network scalability, flexibility, and manageability are key weapons in your war chest to ensure its incumbency and quell user uprisings. Be prepared to increase network capacity incrementally and dramatically when and where users demand it. Make sure the network can adapt to change; make sure it's easy to modify, upgrade, and expand. All of these things are needed to maintain and enhance performance. Finally, do everything in a controlled, responsive, and reliable manner. Management is the most essential design objective; you won't be able to achieve any others, let alone support mission-critical applications and control costs if your network isn't reliable, available, and serviceable (RAS, once again).

8.3.3 Key methods for the network architecture

Our network architecture depends on five methods, of which the most important is centralization. The others are distribution, consolidation, duplication, and segregation. Some of these methods may appear to be contradictory, but, in fact, they are complementary.

Centralization of equipment and services within the confines of a data center is a major feature of our networking strategy. It is the best way to simplify the network and maintain high levels of RAS-just as it's been successful in mainframe environments. For example, we put critical network devices, such as the local routers that make up the backbone network, within the physical confines of a data center and under the control of IT. That way, it's easier to install, monitor, upgrade, and repair critical network components. Obviously, you can't put all network components of a distributed environment into one data center; otherwise it wouldn't be distributed. So, distribute data centers. Even then, unless you are a major control freak and are willing to pay for the facilities and personnel needed to operate hundreds of data centers, you've got to disperse the majority of remote routers, concentrators, and other equipment for your distributed network close to users.

Wait – how can you have centralization and distribution at the same time? There is a very important distinction between distributed versus traditional mainframe centralization: When we talk about centralized networking for distributed computing environments, we talk about control, which is not necessarily where we put equipment. Centralized control means IT has the means to manage any component anywhere on the distributed network, whether those components physically reside in a data center or not. Centralization means we provide the same reliability, availability, and serviceability for distributed environments as we have in the traditional mainframe data center, but we do it over the distributed network. That is the essence of our claim: "The network is the data center!"

In fact, we use centralization and distribution to our advantage. Distribution, by its very nature, puts computing services into the hands of users, where it belongs. By centralizing certain devices in the data center, however, we often can consolidate components and functions.

That leads to savings and manageability, particularly when you must duplicate critical components to afford reliability. For example, reduce the number of routers to manage by consolidating several centralized routers into a single router. Although consolidation enhances manageability, it also leads to problems, such as single points of failure and performance bottlenecks. So our network architecture depends on other complementary methods that appear to be contradictory to enhance RAS.

In addition to centralizing, distributing, and consolidating, we duplicate and segregate, duplicating key components to add redundancy and bypass single points of failure. And we segregate functions to spread utilization across more devices and enhance availability and performance. The network architecture for the New Enterprise depends on an intricate interplay between centralization, distribution, consolidation, duplication, and segregation. The essence of network design is discovering the right balance.

8.3.4 Standards issues

At the heart of most arguments between "open" versus proprietary systems remains the issue of standards. System innovators don't want standards because they dampen change and often lock in a narrow range of solutions. Yet, it's difficult-if not impossible-to implement an enterprise network unless you adhere to some common conventions. The more heterogeneous the systems, the harder it is to manage the networks and (more importantly) to control costs.

Again, production-quality, distributed-computing systems are a balancing act between new technologies and traditional practices. In our view, standards are essential. We depend on them to introduce some order into an otherwise complex heterogeneous network. Fortunately, there are many industry standards that offer an array of solutions. But when it's time to implement your enterprise network, select and adhere to a few standards and choose one product from each. Even though two products are based on the same industry standard, there is no guarantee they are compatible. In our experience, compatibility often is the exception, not the rule.

Today, some of the industry networking standards we recommend include EIA/TIA-568 for cabling and connectors, SNA 3270 terminal-to-program and LU6.2 program-to-program communications for mainframes, and TCP/IP protocols for all other systems. We manage the network through the Simple Network Management Protocol (SNMP) standard. Media access is based on IEEE 802 standards. We use IEEE 802.3 10Base-T Ethernet and IEEE 802.5 Token Ring for the local area networking topology.

Once network standards are defined, it doesn't mean you're finished. New standards evolve as better technologies emerge. You've got to be flexible and take advantage of those new technologies, but be careful to select one for implementation that will avoid compatibility problems. And make sure the various departments and units of your enterprise adhere to those standards. Otherwise, do what we do: Refuse to support those maverick systems.

8.3.5 Survey technologies and trends

To develop a RAS network architecture, it is important to identify and understand the main technologies, issues, and trends. But don't do it as a "cramming" exercise: Education is an ongoing process, and we think it's the most important activity. Remaining well informed means solving problems, taking strategic advantage of emerging technologies, and saving money.

8.3.6 Logical network topology

After analyzing user requirements, setting critical goals, and surveying the technologies, it's time to develop a logical topology for your distributed network architecture. The logical topology revolves around a backbone, the central conduit between major network segments.

What about the remote locations? Subnets can be connected to the backbone network independently of location. This means you can connect the remote subnets in your remote offices as well as the local subnets in the building complex to the backbone. Do this by extending the backbone over a wide geographical area using the WAN.

8.3.6.1 Subnets and workgroups

According to our definition of an enterprise network, each and every connection ("node") on the network has access to all others. Our topology provides the internetworking means for enterprise wide services, such as e-mail, access to warehouse databases, and so on. However, why do we divide the network into smaller segments, particularly since the extra equipment is expensive and the increased complexity makes it more difficult to manage? Small organizations of fewer than 50 nodes can usually do without sub-netting. However, for larger organizations, it's a performance versus complexity trade-off: Sub-netting eases network traffic congestion. Most network-based data and services sharing happens among closely related workers-those workgroups we discussed earlier. Sub-netting isolates local networking traffic to a logical working unit, so it does not contend for network bandwidth with other segments or on the internetwork backbone.

Subnets and workgroups often are assigned along physical or political lines; for example, by department or by corporate division. We take a different tack and decouple the workgroups from the organizational structure and physical network. Workgroups are distinct in their function, not necessarily location, even though their members most likely work close to one another. Networking for workgroups means we have the infrastructure to support dynamic workgroups across the entire enterprise network.

Defining workgroups and assigning users to subnets is not an easy task because workgroups are not static. Members and their tasks in a workgroup change in a dynamic organization; so will their network data traffic patterns frequently change. Workgroup data traffic is not always confined to users in a local area; it may extend across a wide area or several floors in a building complex. For example, a user may be a member of different workgroups for e-mail, file sharing, forms-processing, and mainframe applications. How does this affect our strategy? We separate the physical network from the workgroups. Our strategy is to base the logical network on a dynamic corporate view of workgroups, so any application or service can be accessed by any user, independently of location and time. As you'll see later, this will have a major impact on some of our design decisions.

8.3.7 Management and security

Manageability is the keystone of our networking strategies and a key design objective. We plan and plan and plan for network management right from the outset. It would be difficult and costly to do so later. Our network management strategy is highly proactive. We frequently monitor availability and performance of all the network components so we can anticipate problems and resolve them quickly and efficiently.

Our network management strategy has three key features. First, we manage the network from a central point. This does not mean we centralize all network management processes. We may distribute any process to any point on the network. But we control all processes from a centralized location. Second, all network management processes are based on a common network management platform. The common platform lets us build a centralized network management system by using a variety of third-party products. A network management platform provides a set of functions to collect and display information about the devices connected to a distributed network. We can use these functions directly to manage the network, or we can use them indirectly through network management applications that run on top of the management platform. Network management applications are supplied by a variety of third parties and usually enhance the functionality provided by the network management platform. Most network management applications are designed to run on the common network management platforms, which include SunSoft's SunNet Manager, Hewlett- Packard's OpenView, and IBM's NetView. And third, the network management system is based on a standard-SNMP because it is the de facto network management standard. All network components should be able to communicate with the network management system via SNMP.

We view network security as an important part of network management. With the proper checks and balances, your network, particularly one that supports mission-critical applications, can be secured against inadvertent or malicious acts that can compromise the integrity and confidentiality of your network and your systems, data, and processes. Like all other features in our network architecture, we plan for security from the beginning.

Overall, the prudent objectives for network security are confidentiality, integrity, and availability. Data and systems should be available only to authorized users, and they should not be exposed to accidental or

malicious modification or interruption by users, errors, or failures.

We cannot expect to completely secure the entire network because of the difficulties and costs. We therefore set an acceptable level of risk. It takes some experience to do that, inasmuch as many of the common security countermeasures have weaknesses. The toughest offender (the password "attack") too often succeeds, for example, because it is difficult to force users to secure their passwords properly. It's also well known and easy to configure a notebook computer as a protocol analyzer and monitor data traffic without being detected on an Ethernet or Token Ring network. WANs are particularly susceptible to snooping because you can't easily secure the network "wires." (WANs use many types of channels, such as wires, optical-fiber, microwaves, and satellites.)

In the face of all this vulnerability to errors, failures, and disasters, can we implement a secure distributed network? Definitely, yes. But only if security measures and countermeasures are included as critical parts of the network management architecture and design.

8.4 Designing Switched Networks

This section presents guidelines and configuration examples for using switches in your workgroup networks. It begins with a discussion of the basic 10Base-T Ethernet workgroup network and progresses through various LAN models including high-speed switching and shared media LANs.

8.4.1 Basic Legacy Ethernet Model

8.4.1.1 Description

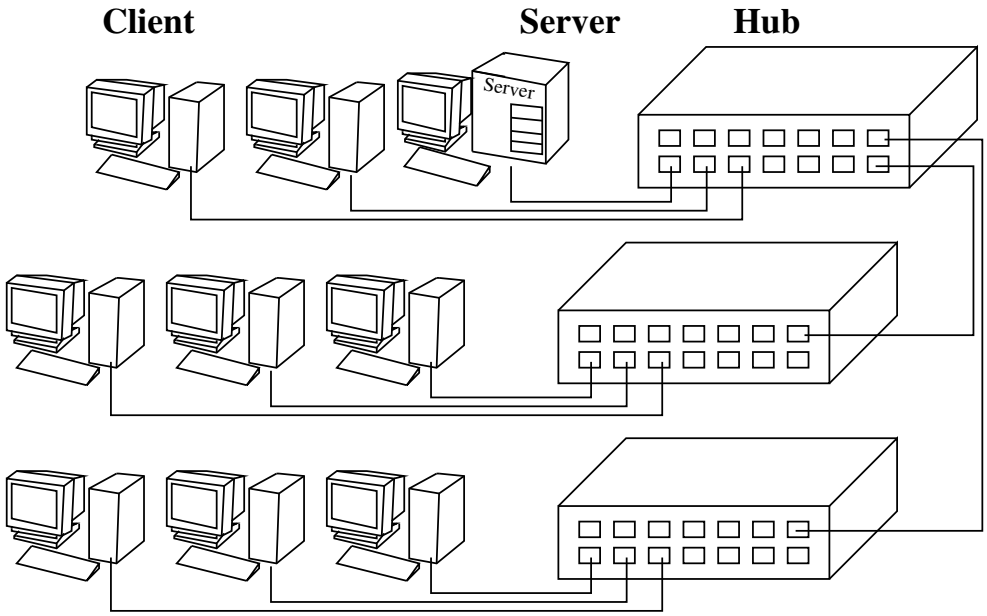


Figure 8.1: Basic Legacy Ethernet

Figure 8.1 shows a moderately sized workgroup with a combination of 80 clients, network printers and one server. The four lower hubs in this picture are connected to the top hub, thus producing a single level of cascading. These hubs can be cascaded with either twisted-pair or fiber-optic cabling.

Legacy Ethernet, shown here as a physical star, is really a bus architecture. with the bus being formed by the repeaters/hubs. If you removed the hubs, you could connect all the clients, servers and printers with the same piece of 50 ohm coaxial cable. If a hub was not used and only cable connected the clients and servers, the bus would be considered passive (it does not require power to run a piece of cable). With

the hub in place, the bus or backbone is considered active. Either way, a legacy Ethernet bus architecture allows for a single half-duplex conversation at a time.

This model is expanded upon throughout this paper to illustrate increases in performance capacity, application growth, network expansion, and workstation count increases.

8.4.1.2 Benefits

This model is the most simple and inexpensive. It is also the most simple and proven way to build an Ethernet LAN.

The legacy Ethernet model is well understood throughout the industry.

8.4.1.3 Considerations

Legacy Ethernet is not fully capable of supporting many multi-media clients, or a mixture of multi-media and data base access requirements due to the CSMA/CD (non-deterministic) access method of Ethernet/802.3.

8.4.1.4 Performance

With legacy Ethernet, CSMA/CD is the access method. Because of this access method, actual network capacity reaches a maximum at about 60% utilization when there are many nodes attempting to access the network at approximately the same time. This means that Ethernet with CSMA/CD measures about 6 Mb/s maximum capacity. This does not mean that the actual data is transmitted at 6 Mb/s, it means that if many nodes are heavily utilizing the LAN, you will only get a maximum capacity for the entire shared collision domain of roughly 6 Mb/s. Thus, each node shares 6 Mb/s, not 10 Mb/s of bandwidth. Of course, if node count and demands for heavy access to the media decrease, the available bandwidth per node increases.

8.4.2 The Distributed Server LAN Model

As a customer's needs grow beyond the ability of a 10-Mb/s shared media Ethernet LAN, a switch can be deployed to help increase inter-segment capacity, isolate workgroups, and generally increase performance to global resources. As more applications are required, the need for more servers will also arise. As more clients are added to the network, additional servers may be added to handle the additional demands for redundancy and/or distribution of liability.

8.4.2.1 Description

Figure 8.2 shows a decentralized approach to switching design. Unlike a server farm, heavily used resources are dispersed throughout the site. This can be a valid chosen design, or something that has grown over time. If this network has grown over time, the network resources and clients were likely placed at key locations as they were deployed or as a result of protocol separation/segmentation. Figure 8.2 shows the switch at the top-of-stack. Each workgroup pictured may actually be running a separate layer 3 protocol like IP, IPX or AppleTalk; or a bridgeable protocol such as NETBIOS, NETBEUI, or Dec LAT. If you have many separate workgroups and want to bring them together into a collapsed backbone, then a switch would be placed at the top of the stack as illustrated. Without this switch, independent workgroups would remain.

Note that in a switched media network, network printers are considered as clients unless they are a heavily utilized high-capacity printer or plotter. Printers generally receive data. Data transmission is typically reserved for control and responding to SNMP queries. Full-duplex links to these types of devices are not required. Scanners are just the opposite.

However, traffic flows to the printer may differ based on the network platform used (such as UNIX/IP, NT/IP, NT/NETBEUI, Novell 3.x IPX, Novell 4.x IPX, or AppleTalk). For example, a printer in a UNIX/IP environment may receive data from any number of client workstations. It is important to understand your applications, environment, and traffic flow patterns before you design your network.

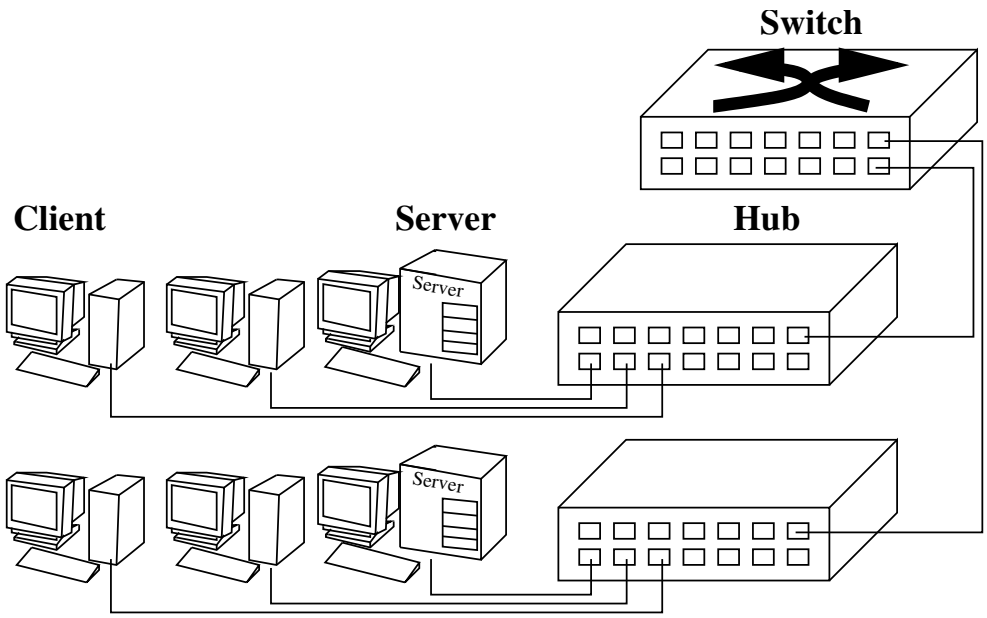


Figure 8.2: Distributed Server Model

8.4.2.2 Benefits

This model is useful for reducing congestion on legacy Ethernet. By isolating or segmenting the shared media workgroups, not as many nodes are contending for the same bandwidth.

Since the technology does not change in this model, you will likely be able to use all existing wiring, including any 10Base-5 coaxial cable, 10Base-2 coaxial cable, or 10Base-F fiber-optic cable.

Workgroups (lab, manufacturing, marketing, finance, etc.), applications, protocols can all be segmented for any number of technical or logistical reasons.

8.4.2.3 Rules of Thumb

Although the backbone capacity in this design is greater than that of a single shared media workgroup, inter-segment traffic through the switch traffic should be kept to a minimum. Use the 80/20 rule of 80 percent traffic is from the client to local workgroup server (intra-segment), and 20 percent is traffic out to other workgroup servers (inter-segment).

A single shared media 10-Mb/s network that is not congested will not see a benefit from this collapsed backbone design. This design is only useful for reducing congestion.

8.4.2.4 Performance

Performance on each workgroup will increase if the network was previously a single shared media network experiencing problems with congestion.

8.4.3 Server Farm LAN Model

8.4.3.1 Description

Figure 8.3 shows a centralized approach, often referred to as a server farm where all heavily used resources are located a central location. The switch at the top of the stack is considered a collapsed backbone, since many individual workgroup LANs are brought together into one active component. In this example, the individual workgroups may have previously been cascaded off the top workgroup hub. Each server is connected to the switch via a high-speed link or fat-pipe.

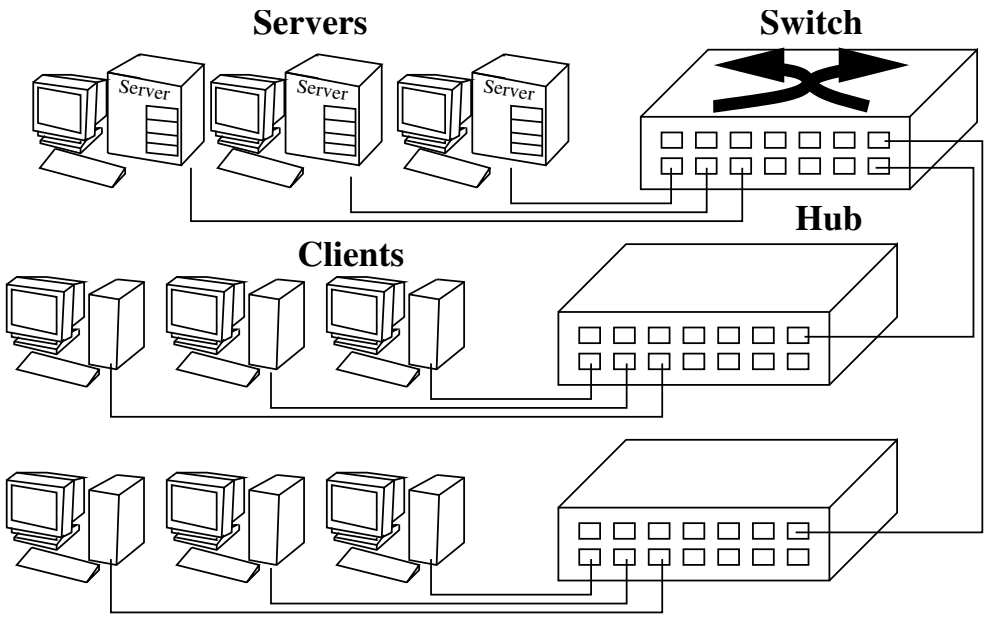


Figure 8.3: Server Farm model

8.4.3.2 Benefits

Generally, the server farm design with fat-pipes connecting the servers, out performs the distributed server model and is the preferred switching network design.

This design provides physical security for the servers. It is easier to manage and maintain the servers in this environment.

From a logistical viewpoint, you can keep all client LAN adapter cards if this is an upgrade from an existing shared Ethernet.

You can use all current workgroup wiring, including any 10Base-5 coaxial cable, 10Base-2 coaxial cable, or 10Base-FL fiber-optic cable. Remember, all we are doing is linking an Ethernet hub up to a 10/100 switch. The workgroups can stay the same.

If security of additional performance is required, some switches support MAC-to-port filters to limit access of server traffic to specified clients in a server farm topology.

8.4.3.3 Rules of Thumb

Demand (client workgroup ports), should equal resource (server ports) when maximizing the number of possible simultaneous conversations through the switch. This rule holds at almost one-to-one for a 10/10 or 100/100 switching, but since the servers are on fat pipes the equation is modified slightly. The goal is an efficient design that maximizes capacity and throughput, yet without causing undue congestion. A more effective and useful rule to follow would be no more than fourteen 10-Mb/s demand ports to a single 100-Mb/s resource port. For example, an average Ethernet packet size of 512 bytes, for a 100 Mb/s technology (including overhead) is about 20,000 PPS. Each 10 Mb/s Ethernet would be considered to have 60 percent cross switch traffic, or 1410 PPS. Divide 20,000 by 1410 to obtain the port count of 14.2 rounded down to 14.

Resources should be fed with high-speed links, the links to the servers should be of greater capacity than the workgroup links. In fact, without a server fat-pipe, performance through a 10-Mb/s switch could be worse than in a 10-Mb/s shared network. This is because in a shared network, the end node simply holds or defers transmission of the data until it gains access to the media, keeping the delay handling at the lowest levels. In a switched network, the higher layers of the end node's protocol may be called upon to retransmit in the event of dropped packets by the switch.

8.4.3.4 Considerations

Consideration should be given to the high-speed uplink, fat-pipe technology chosen for switch-to-server communications. If a server is typically always transmitting (for instance, 80 percent transmit and 20 percent receive, then simply making a 10-Mb/s Ethernet port a full-duplex port would not yield 20 Mb/s, but more like 12 Mb/s. The same is true if a 100Base-T full-duplex port is chosen. It would be 120 Mb/s, not 200 Mb/s. This should not be a concern with switch-to-switch communication, unless it has been proven through traffic analysis that the flow of traffic between the switches is not equal.

8.4.4 The Desktop Model

8.4.4.1 Description

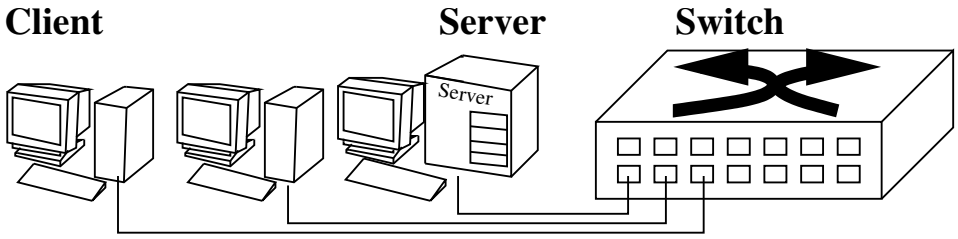


Figure 8.4: The Switching Workgroup

Figure 8.4 shows a small workgroup with a node or micro-segment switch as the active networking component. The clients are connecting to the switch via their current 10Base-T cards and cabling. The server is connecting to the switch with a high-speed link or fat-pipe. Although this figure shows directly connected clients, low port count hubs could replace each of the clients in the figure.

Note that this type of switched workgroup could also be connected with other workgroups.

8.4.4.2 Benefits

This is an effective low-cost upgrade solution for the small but demanding LAN. Clients will not need to upgrade their current 10Base-T cards to accommodate a new networking technology.

This model provides significant performance improvements due to high speed link to server.

8.4.4.3 Rules of Thumb

The server fat-pipe should be of a greater speed than the aggregate speed of the client ports.

8.4.5 Switch-to-Switch Links

Your switched network topology will differ depending on the media type chosen for the switch-to-switch link. The following provides a list of the different switch-to-switch links and the maximum distance allowed between the switches using that technology.

8.4.5.1 Rules of Thumb

It is a good idea to keep inter-switch links passive. To enhance reliability, it is recommended that switch-to-switch communication does not go through other active components, unless there is a level of redundancy in those components.

Different media types have different distance limitations for the links connecting the switches. Below are the maximum distances achievable within some of the various media types without any active device in the link:

Inter-Switch Link Technology	Max distance between switches
802.12 100Base-VG Cat 3 UTP	100 meters
802.12 100Base-VG Cat 5 UTP	200 meters
802.12 100Base-VG Multi-mode Fiber	2000 meters
802.3 Clause 25 100Base-TX	100 meters
802.3 Clause 23 100Base-T4	100 meters
802.3 Clause 26 100Base-FX	412 meters
802.3 100Base-FX (Full Duplex)	2000 meters
802.3 100Base-FX (FD with single mode)	20000 meters
FDDI	2000 meters
802.3 10Base-T	100 meters

8.4.5.2 Performance

The best media type to choose for switch to switch links will vary dependent upon topology restraints, and numerous other factors. When considering purely speed, the switch-to-switch, and for that matter the server fat pipe, links would be ranked as follows:

1. Full Duplex 100Base-T: Excellent performance for single switch-port to switch-port communication.
2. 100VG AnyLAN: Excellent performance for multiple switch connectivity. Lower cost than FDDI.
3. ATM: Excellent performance, but will be more costly than other solutions.
4. Half-Duplex 100Base-T: Good performance for single node-to-node communication.
5. FDDI: Excellent performance for multiple switch connectivity.
6. Full Duplex 10Base-T: Last resort but will work if you are only switching 10 Mbps networks.

8.4.6 The 100-Mb/s Switching Backbones Model

8.4.6.1 Description

This model shows how switching is used to interconnect workgroup networks of different technologies. Figure 8.5 shows a 10-Mb/s distributed server switched workgroup and a legacy Ethernet shared workgroup, connected with a switching backbone.

8.4.6.2 Benefits

This model provides segmentation between the various workgroups. Each workgroup uses the technology required for their applications.

It provides ease of migration from 10-Mb/s technology to 100Base-T technologies.

This model enables communication between previously separated workgroups. Even though each is running a different media type, all clients can communicate with all resources.

This model also provides port trunking depending on the type of switches used. Port trunking allows you to link two switches together with multiple links. For example, you can link two switches together boosting your switch-to-switch performance by doubling the throughput. Note that both switches must support port trunking to use this feature.

8.4.6.3 Rules of Thumb

The workgroup on the bottom is a distributed server switching design. The link between the top-of-stack switch shown in this workgroup and the segment switch above it should be a high-speed link such as 100Base-T, 100Base-T full duplex, or FDDI.

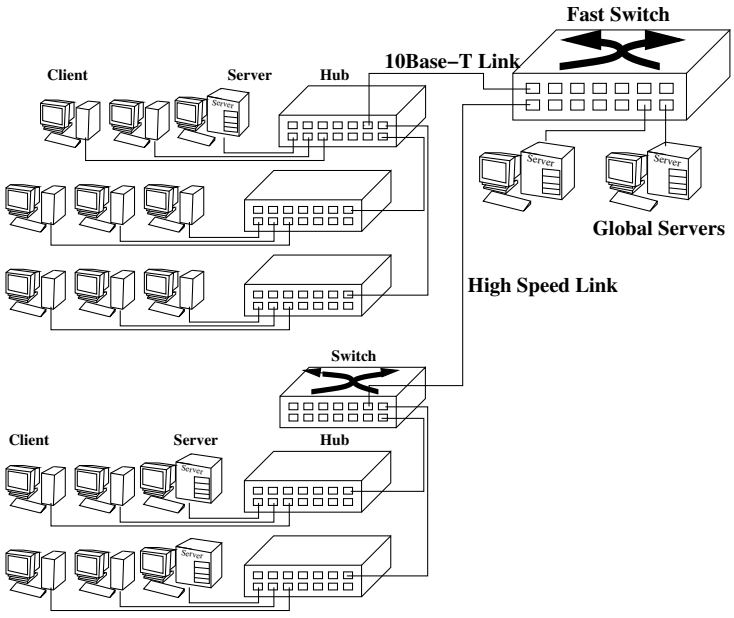


Figure 8.5: LANs using switching backbones

8.4.6.4 Considerations

In this model, it is still best to keep most of the traffic within the local workgroup. If the customer's applications demand more cross-switch traffic to other resources, it will be best to build a server farm to those global resources, using the server farm model shown earlier.

8.4.7 Shared Media LANs Using FDDI Backbones

FDDI has gained wide acceptance as a high-speed 100Mbps shared media backbone. Fiber-optic cabling is immune to noise and is light weight. Architecturally FDDI also has excellent redundancy characteristics. It is also well suited for building-to-building campus and high rise cable plants.

8.4.7.1 Description

Figure 8.6 shows FDDI as a shared media 100Mbps backbone.

8.4.7.2 Benefits

The basic benefits of FDDI are redundancy, 100-Mb/s data rate, and the durability of the fiber itself. Since FDDI is a mature technology, FDDI interfaces are very common today. This allows for a great deal of flexibility in connectivity. Many vendors support FDDI for the interfaces on their network devices. Often it is the backbone of choice in a heterogeneous collection of server and client platforms.

Segmentation between various workgroups is done with the local top-of-stack switches.

8.4.7.3 Rules of Thumb

If you are using FDDI to connect switched 10-Mb/s workgroups, and have no global servers on the ring, then you will want to follow the 80/20 rule. Through the FDDI ring (inter-workgroup) traffic should be designed to be a minimum. 80 percent traffic is client to local workgroup server (intra-workgroup), and 20 percent is traffic out to other workgroup servers (inter-workgroup).

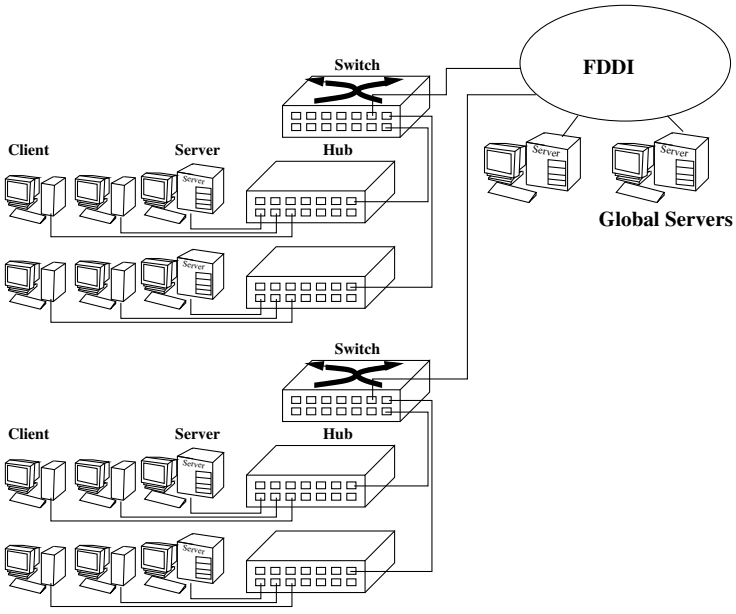


Figure 8.6: LANs using FDDI backbone

8.4.7.4 Considerations

The biggest potentially negative factor in an FDDI design may simply be cost. The costs for network adapters for servers in the server farm, and fiber installation costs can be high. Admittedly these costs have dropped in recent years, but it may still be high in your area.

8.4.8 Switched Networks Using Redundant Paths Model

8.4.8.1 Description

Figure 8.7 shows how redundant paths are implemented in a switched network. This design requires that your switch support the 802.1d spanning tree specification. 802.1d specifies how the switches/bridges will communicate among themselves to close off one of the paths or open the loop. 802.1d also specifies how the path will be reestablished in the event that another path is broken by link or switch failure. The model shows a physical look, how the wiring connects the switches. When spanning tree is implemented, this diagram would show one of the switch-to-switch links removed. Although the model shows straight-line links between the switches, any shared media, active or passive, could be used in place of a direct link.

8.4.8.2 Benefits

With this design, there are two physical paths from any workgroup to any workgroup.

8.4.8.3 Rules of Thumb

802.1 specifies that there be no more than seven bridges (or switches) between any two end nodes, for spanning tree protocol support.

8.4.8.4 Considerations

In the above figure, with only one loop is created, you need to consider the greatest number of hops between workgroups. In this example, a client in one workgroup would need to cross through four switches

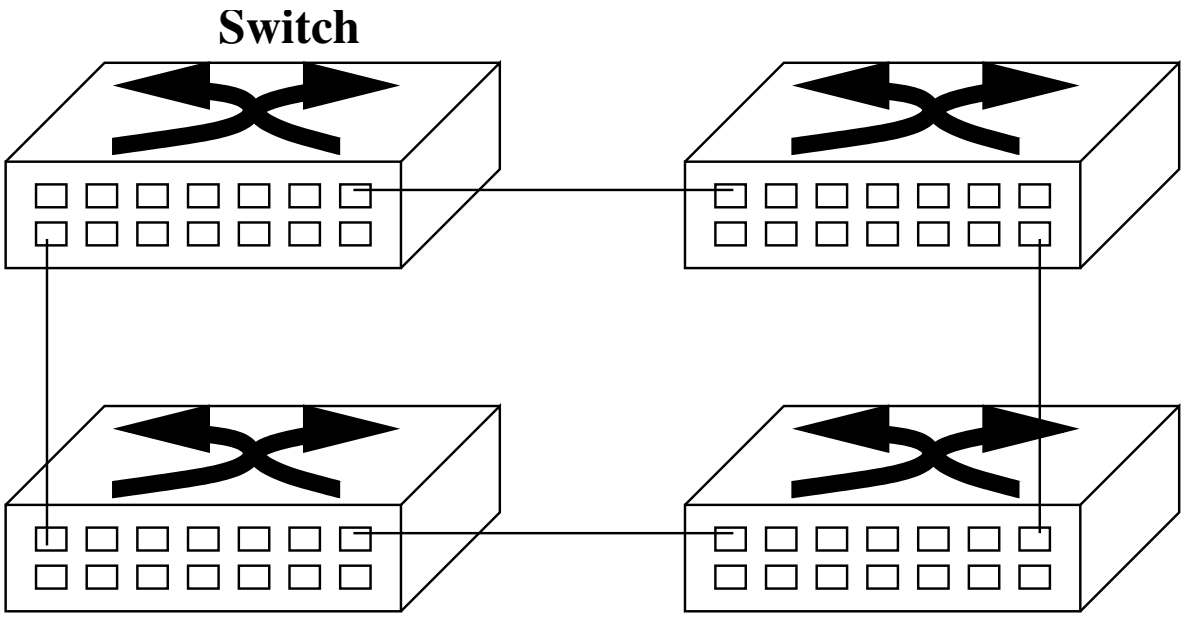


Figure 8.7: Redundant Paths in a Switched Network

to communicate with the furthest workgroup. A better way to build redundancy and maintain higher performance is to create more links between the switches. However, to implement this type of design, you will use more links for redundancy, rather than for workgroup connections.

8.4.8.5 Performance

The largest concern with this model is the latency of each switch. If your needs do not require all clients talk with all servers equally, then the model shown in Figure 8.7 will not impair performance.

8.5 Designing with Gigabit Ethernet

This section discusses the various topologies in which Gigabit Ethernet may be used. Gigabit Ethernet is essentially a "campus technology", that is, for use as a backbone in a campus-wide network. It will be used between routers, switches and hubs. It can also be used to connect servers, server farms (a number of server machines bundled together), and powerful workstations.

Essentially, four types of hardware are needed to upgrade an exiting Ethernet/Fast Ethernet network to Gigabit Ethernet :

1. Gigabit Ethernet Network Interface Cards (NICs)

2. Aggregating switches that connect a number of Fast Ethernet segments to Gigabit Ethernet

3. Gigabit Ethernet switches

4. Gigabit Ethernet repeaters (or Buffered Distributors or Gigabit Ethernet hubs)

The five likely upgrade scenarios are given below.

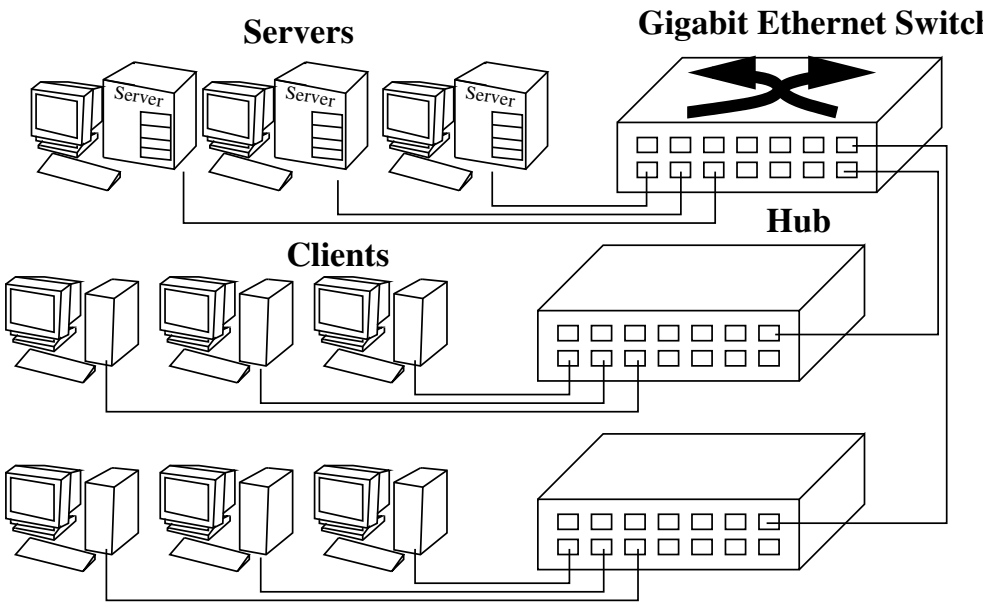


Figure 8.8: Upgrading server switch to Gigabit Ethernet.

8.5.1 Upgrading server-switch connections

Most networks have centralized file servers and compute servers A server gets requests from a large number of clients. Therefore, it needs more bandwidth. Connecting servers to switches with Gigabit Ethernet, as shown in Figure 8.8 will help achieve high speed access to servers. This is perhaps the simplest way of taking advantage of Gigabit Ethernet.

8.5.2 Upgrading switch-switch connections

Another simple upgrade involves upgrading links between Fast Ethernet switches to Gigabit Ethernet links between 100/1000 Mbps switches, as shown in Figure 8.9.

8.5.3 Upgrading a Fast Ethernet backbone

A Fast Ethernet backbone switch aggregates multiple 10/100 Mbps switches. It can be upgraded to a Gigabit Ethernet switch which supports multiple 100/1000 Mbps switches as well as routers and hubs which have Gigabit Ethernet interfaces. Once the backbone has been upgraded, high performance servers can be connected directly to the backbone, as shown in Figure 8.10. This will substantially increase throughput for applications which require high bandwidth.

8.5.4 Upgrading a Shared FDDI Backbone

Fiber Distributed Data Interface (FDDI) is a common campus or building backbone technology. An FDDI backbone can be upgraded by replacing FDDI concentrators or Ethernet-to-FDDI routers by a Gigabit Ethernet switch or repeater.

8.5.5 Upgrading High Performance Workstations

As workstations get more and more powerful, higher bandwidth network connections are required for the workstations. Current high-end PCs have buses which can pump out more than 1000 Mbps. Gigabit Ethernet can be used to connect such high speed machines.

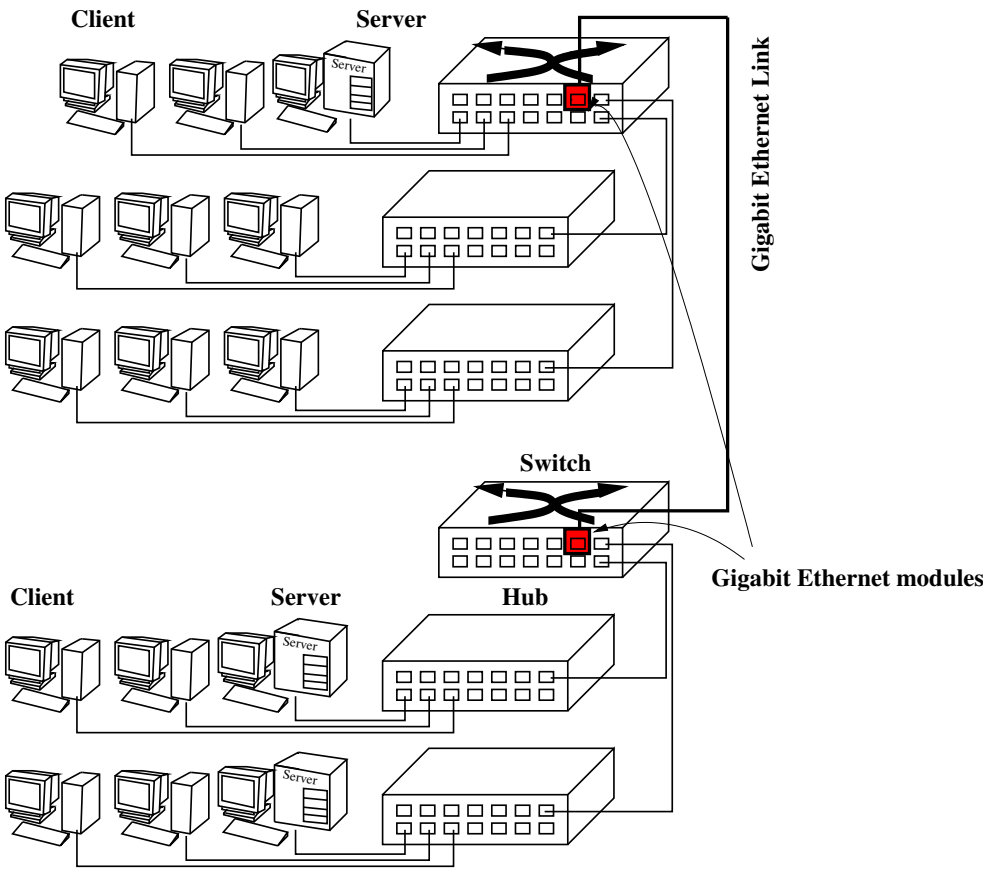


Figure 8.9: Adding Gigabit Ethernet links.

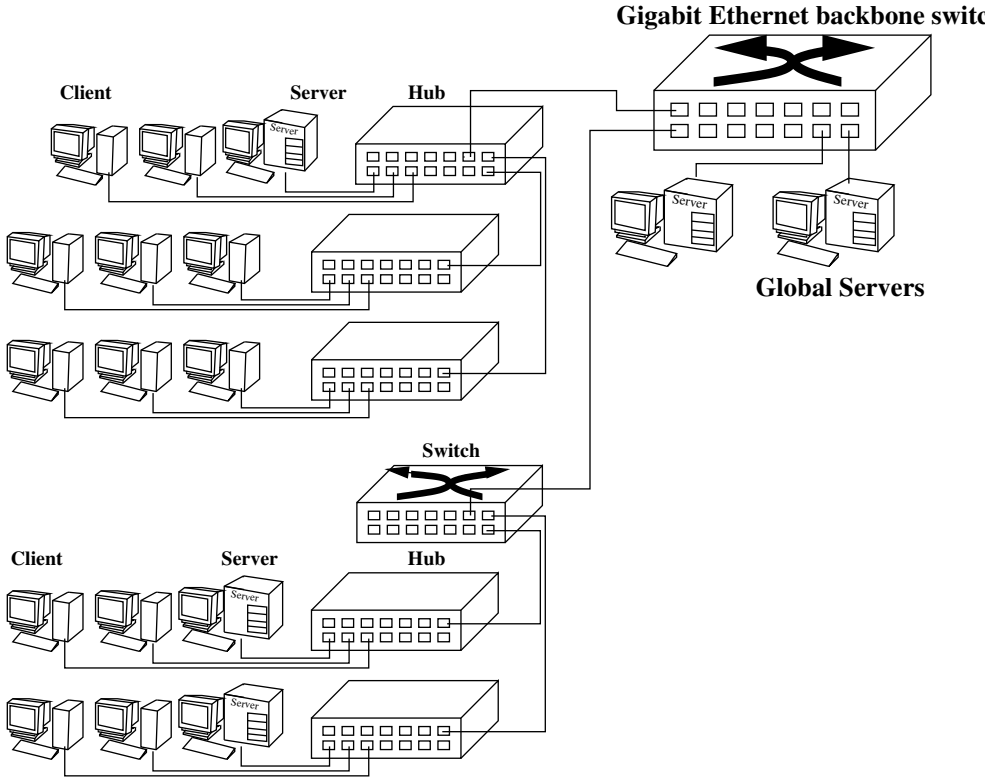


Figure 8.10: Gigabit Ethernet backbone.

Chapter 9

Network Analysis

9.1 Chapter Structure

9.1.1 Outcomes

1. Be able to analyze and critically comment on network models.
2. Model computer networks to provide and analyze data relating to performance issues.
3. Ability to use a standardized representation to represent the behaviour of a computer network and network protocol.
4. Ability to evaluate alternative scenarios when solving problems involving computer networks.

9.1.2 Objectives

This chapter:

1. Introduces Petri Nets as a standardized modelling strategy.
2. Describes analysis techniques for automatically extracting properties of a Petri Net model.
3. Considers interpretation of properties of the Petri Net model in terms of the system being modelled.
4. Describes simulation as a mechanism for the analysis of Petri Net models and other event based modelling techniques.
5. Describes the use of simulation for performance evaluation purposes, and discusses issues involved when taking measurements from a simulation.

9.2 Petri Nets

Petri Nets were introduced by C.A.Petri in the early 1960s as a mathematical tool for modelling distributed systems and, in particular, notions of concurrency, non-determinism, communication and synchronization. There are many varieties of Petri Nets from black and white nets, which are conceptually simple and straightforward to analyze, to more complex nets such as coloured nets which allow the modelling of complex systems.

Definition: Petri Nets A Petri Net PN is an algebraic structure (P,T,I,O) composed of:

- A finite set $P = \{p_1, p_2, .., p_n\}$ whose elements are called Places,
- A finite Set $T = \{t_1, t_2, .., t_m\}$ whose elements are called Transitions,

- A Transition Input Function - I. The I function is a function that maps each transition $t_i \in T$ to a multiset of P.
- A Transition Output Function - O. The O function is also a function that maps each transition $t_i \in T$ to a multiset of P.

Definition: Markings/Tokens/Initial Marking A marking in a Petri Net $PN(P,T,I,O)$ is a function μ, that maps every place into a natural number. If for a given marking μ, $\mu(p_i) = x$, then it is said that the place p_i holds x tokens at the marking μ.
A special marking, denoted by μ_0, will be called the Initial Marking.

Definition Enabled/Fireable Transitions at a marking μ. At a given marking μ, if for any $t_i \in T$ such that $\mu(p) \geq \#[p,I(t_i)], \forall p \in P$ then t_i is said to be enabled by the marking μ. Here $\#[p,I(t_i)]$ denotes the number of occurrences of place p in the multiset $I(t_i)$. Lets denote the set of all enabled transitions at a given marking μ by $EN(\mu)$.

In conventional Petri Nets every enabled transition may fire. This is not always true for other kind of Petri Nets, particularly the time(d) ones.

If we denote by $F(\mu)$ the set of all fireable transitions at a given marking μ, then for conventional Petri Nets $F(\mu) = EN(\mu)$

Definition Firing of a Fireable Transition The firing of any enabled transition, t_i, at marking μ, causes the change of the marking μ to a new marking μ' as follows: $\forall p \in P, \mu'(p) = \mu(p) - \#[p,I(t_i)] + \#[p,O(t_i)]$
Where: $\#[p,I(t_i)]$ and $\#[p,O(t_i)]$ denotes the number of occurrences of place p in the multiset $I(t_i)$ and in the multiset $O(t_i)$ respectively.
In other words, the new marking μ', for each place p, is equal to the old number of tokens in that place, minus the number of occurrences of p in the Input Function of t_i, plus the number of occurrences of the place p in the Output Function of transition t_i.

9.2.1 Graphical Representation

The algebraic structure of a Petri Net $PN(P,T,I,O)$ may be represented graphically. In this graphical representation, a Petri Net will be represented by a bipartite graph, where:

- every place will be represented by a circle
- every transition will be represented by a little rectangle
- the function I will be represented by directed arcs linking every $p \in I(t_i)$ to the transition t_i. These arcs are called input arcs to the transition t_i
- the function O will be represented by directed arcs linking each transition t_i to every $p \in O(t_i)$. In analogy with the input arcs, these last arcs are called output arcs to the transition t_i

If the number of occurrences of p in $I(t_i)$ or $O(t_i)$ is greater than one, say n, then instead of representing each occurrence by a different arc, it is recommended to represent all of them by a single arc labeled by n.

Some authors represent transitions graphically by little squares instead of little rectangles.

9.2.2 Modeling discrete systems with Petri Nets

When modeling a discrete system with a Petri Net, partial states of the system are represented by places. Whether the system is in a particular partial state or not is represented by the presence/absence of a dot in the place representing this partial state. Events are represented by the transitions. Conditions allowing an event to occur are represented by the input arcs to the associated transition of this event. These are normally called pre-conditions. The input places of these arcs represent the combination of the several partial states that must be valid in order that the event represented by the transition occurs. After the occurrence of an

event (firing of an enabled transition) a new set of partial states will be valid. These are called the post conditions and are represented by the output arcs of the fired transition.

To create a Petri Net:

1. A partial state is the state of a component of the system.
2. Identify the components of the system. List all the components.
3. Identify the states of each component. List all these partial states, and label them ($P_1..P_n$).
4. Events are represented by the transitions. List all these events, and label them ($T_1..T_m$).
5. Conditions allowing an event to occur are represented by the input arcs to the associated transition of this event. These are normally called pre-conditions. List all these pre-conditions by placing the label for the input partial state next to the corresponding events.
6. The input places of these arcs represent the combination of the several partial states that must be valid in order that the event represented by the transition occurs.
7. After the occurrence of an event (firing of an enabled transition) a new set of partial states will be valid. These are called the post conditions and are represented by the output arcs of the fired transition. List all the post-conditions by placing the label for the output partial state next to the corresponding events.
8. Convert your lists to the diagram by replacing partial states with places (circles), events with transitions (bars), pre-conditions with input arcs and post-conditions with output arcs. Label all places and transitions.
9. The initial marking represents the initial state of each component of the system. Draw in the initial marking of the Petri Net.
10. The tokens may be used to represent a value associated with a partial state (such a partial state where the part (e.g. bank) has n tokens (or n coins), rather than just a partial state which is active or not (e.g. bank has money)).
11. The number of input and output arcs are used to ensure that the number of tokens in a particular state has meaning, as tokens move between states.

9.2.2.1 An example of a graphical representation of a Petri Net

The graph in Figure 9.1 is an example of a Petri Net that models a spooler system consisting of 4 computers and a single printer.

At the initial marking, shown in the above graph, there are 4 enabled transitions (t1, t2, t3 and t4) that represents the fact that any one of the four computers is ready to produce a document to be printed and that will be put in the spooler buffer (place p9).

In a Petri Net, there is not any information about when any enabled transition will fire.

So, at any time any computer may produce a document to be printed. Once a document is produced, the computer is free to continue its work and, if it is the case, to produce another document to be printed.

It can be noticed that many different sequences of events may occur, making clear the parallel nature of this system and the modeling power of a Petri Net.

9.2.3 Categories of Petri Nets

9.2.3.1 Black and White Nets

A simple (black and white) Petri Net is a digraph with nodes which may be places (drawn as circles) or transitions (drawn as rectangle or lines). Edges can connect places to transitions (known as input arcs, and the corresponding places known as input places) or transitions to places (known as output arcs, and the

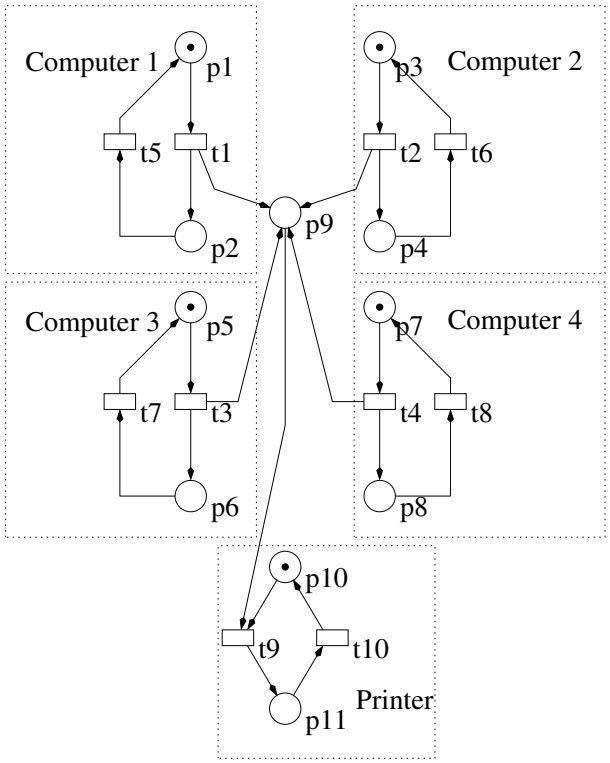


Figure 9.1: Example Petri Net

corresponding places known as output places). A Petri Net can be marked by indicating the tokens which are contained in each place at a point in time (drawn as dots). If the input places of a transition all contain (at least) one token, then the transition is eligible for firing. If it does fire, then one token is removed from each of its input places and one token is added to each of its output places. A Petri Net is executed by establishing an initial marking and then, at each subsequent cycle, choosing a set of eligible transitions for firing.

We note that the ability of a transition to fire is determined solely by local conditions, namely the presence of tokens in the adjacent input places. This locality of reference is a desirable feature in modelling concurrent systems.

9.2.3.2 Coloured Nets

Black and white nets are very useful but limited in their modelling power. A common extension is known as coloured Petri Nets where the tokens are differentiated by colours which may be arbitrary data values. The firing of transitions is then made dependent not just on the availability of a token at the input place, but the availability of an appropriately coloured token. Similarly, the output of a transition is not just a token but a specifically coloured token. The notation of coloured nets is much more concise than black and white nets, and thereby avoids a lot of the duplication which is typical of black and white nets.

9.2.3.3 Time, Timed and Stochastic Nets

Petri Nets can also be used to simulate timed processes by attaching delays to transitions or to places. Conventionally, one of these alternatives is chosen, since either can be modeled in terms of the other. If the delays are randomly distributed (as with Poisson processes), the nets are called stochastic nets and are analyzed with the techniques of Markov processes. If the delays are deterministically distributed, then the nets are referred to as timed nets.

- Stochastic Petri Nets (SPNs) arise when we add stochastic extensions to some pure Petri Net formalism, e.g. to place/transition Petri Nets (PTPNs) or coloured Petri Nets. The classical approach, is to use PTPNs, which offer the easiest ways of analysis. Time extensions of PTPNs can be defined in such a way that every timed transition has a timer, which after having been started with some value (usually derived from a probabilistic distribution) decrements as long as the transition is enabled and the transition fires when the timer reaches zero. Three basic memory policies specifying the way of keeping track of the past are introduced: re-sampling (transition timers are restarted after every firing), enabling memory (only disabled transition timers are restarted) and age memory (timers are restarted only when they reach zero). In the case of multiple enabling we distinguish between single server and multiple server transition semantics. Several classes of SPN are then distinguished according to the possible time distributions.

- Generalized stochastic Petri Nets (GSPNs) allow timed transitions with exponentially distributed delays as well as immediate transitions. Transition priorities are used to avoid confusion and undesirable conflicts. The lowest priority level is reserved for timed transitions. Immediate transition weights say how transitions from extended conflict sets should be fired. GSPNs are a very interesting class of SPNs because every GSPN has an underlying continuous-time Markov process.

- Deterministic and stochastic Petri Nets (DSPNs) extend GSPNs in the sense of allowing transitions with deterministic delays. DSPNs do not belong to the class of Markovian SPNs but their steady-state analysis is still possible, provided that at most one deterministically timed transition can be enabled at a time. Therefore DSPNs do not allow more deterministic transitions to be concurrently enabled. This is not the case of concurrent DSPNs, which can be analyzed by means of steady state approximation. Analysis methods have also been proposed for the class of extended DSPNs where exponentially timed transitions can be used with a restriction that at most one non-exponentially timed transition can be enabled at a time. Exponential distribution covers many very well known distributions, e.g. uniform or triangular distribution.

- A Timed Petri Net is a pair (PN,τ), where PN is a conventional Petri Net (P,T,I,O) and τ is a function which associates a non-negative real number to each transition $t_i \in T$, known as the firing duration of transition t_i.
  Transitions in a Timed Petri Net are enabled by a marking μ the same way as a conventional Petri Net. The firing of an enabled transition t provokes a change of markings in two steps: first, at the moment t is fired, the marking decreases: $\forall p \in P, \mu'(p) = \mu(p) - \#[p, I(t)]$ where $\mu'(p)$ is the resulting marking at p after the first step and $\#[p, I(t)]$ is the occurrence number of the place p in the input of transition t.
  The second step occurs $\tau(t)$ time units afterward and provokes an increase of the markings as follows: $\forall p \in P, \mu''(p) = \mu'(p) + \#[p, O(t)]$ where $\mu''(p)$ is the final marking for p after the second step and $\#[p, O(t)]$ is the occurrence number of the place p in the output of transition t.
  The final marking after firing a transition t is exactly the same as in an associated conventional Petri Net without time parameters.
  Transitions in a Timed Petri Net must fire as soon as they are enabled. This may cause some ambiguities when an enabled transition is disabled by the firing of another one.

- A Time Petri Net is composed by a pair $(PN,\oslash)$, where PN is a conventional Petri Net (P,T,I,O) and is a function which associates an interval of non-negative real numbers $[a_i,b_i]$ to each transition $t_i \in T$. This interval is named Static Firing Interval of the transition t_i. There are no restrictions to the upper and lower limits of this interval, except the fact that $a_i \leq b_i$. This means that a_i can be zero and b_i can be infinite - a_i is named Static Earliest Firing Time and b_i is named Static Latest Firing Time.
  Transitions are enabled the same way as in conventional Petri Nets, but the firing of an enabled transition t will only happen in a time $\oslash(t)$ within limits defined by its Static Interval, relative to the moment t was enabled.
  The firing of a transition in a Time Petri Net is instantaneous and has the same effect as in a conventional Petri Net.

9.2.4 Analysis of Petri Nets

Once a Petri Net has been drawn, it is important to be able to analyze the net to determine what sort of properties it has in order to determine if the design will be feasible for a particular application. The first step of analysis usually consists of forming a reachability tree.

The reachability tree begins with the initial marking. From this marking, a new marking is created for each enabled transition from that marking, and the process repeats. Once a marking is found that is already present in the tree, it is marked and that marking need not be added to the tree a second time. This is known as a duplicate node. Once all possible markings have been found, the tree is complete. The completed reachability tree shows all of the possible markings that can be obtained from the initial marking given every possible sequence of transition firings.

Once the reachability tree is constructed, the tree can be analyzed in order to determine specific properties of the Petri Net. Some of the more important properties that can be determined include safeness, boundedness, liveness, and conservativeness.

The property of safeness can be determined for both individual places and for the entire net. A place is said to be safe if, for all possible markings, the number of tokens in that place never exceeds one. The Petri Net is declared safe if all of the places in the net are safe.

The property of boundedness can also be determined for individual places and for the entire Petri Net. The boundedness property is actual a more general form of the safeness property. A place is said to be k-bounded if, for all possible markings, the number of tokens in that place never exceeds k. A Petri Net is k-bounded if, for all possible markings, the number of tokens in any individual place in the net never exceeds k. Both the safe and bound properties for Petri Nets are important in the field of engineering because they help determine what size buffers, counters, etc. are needed in order to implement the design. For instance, if all of the places are safe, it would be possible to implement these conditions with a boolean variable. If a place is representative of a buffer, and is 16-bounded, then the designer knows to use a buffer of size 16.

The third property, liveness, is one of the most important properties of the Petri Net for most applications. The liveness property encapsulates the concept of a system which will be able to run continuously, i.e. a system which does not deadlock, which is an important property when modeling operating systems, communication protocols, computer programs, and just about any safety critical system. Using the most general definition of liveness, a Petri Net is considered live if, for all possible markings, there is always a transition enabled. There have been, however, many other concepts developed in the area of liveness, some of which define different levels of liveness for individual transitions.

The last property, conservativeness, is associated with the total number of tokens within the Petri Net. A Petri Net is said to be strictly conservative if, for all possible markings, the total number of tokens in the Petri Net always remains constant. A Petri Net can also be conservative with respect to a particular weighting vector which can be defined for a Petri Net. This would allow, for instance, the total number of tokens to increase by some number for certain markings, as long as they were reduced by that same number at a later marking.

In summary:

1. Construct the reachability tree.
   (a) The reachability tree begins with the initial marking.
   (b) A new marking is created for *every* enabled transition from that marking.
   (c) These new markings are added to the tree, and linked to their originating node.
   (d) The process repeats.
   (e) Once a marking is found that is *already present in the tree*, that marking need not be added to the tree a second time - instead a cycle is created in the tree.
   (f) Once all possible markings have been found, the tree is complete.
   (g) The completed reachability tree shows all of the possible markings that can be obtained from the initial marking given every possible sequence of transition firings.

2. Safeness can be determined for both individual places and for the entire net.
   (a) A place is said to be safe if, for all possible markings, the number of tokens in that place never exceeds one.
   (b) The Petri Net is declared safe if all of the places in the net are safe.
3. Boundedness can be determined for individual places and for the entire Petri Net.
   (a) A place is said to be k-bounded if, for all possible markings, the number of tokens in that place never exceeds k.
   (b) A Petri Net is k-bounded if, for all possible markings, the number of tokens in any individual place in the net never exceeds k.
4. Liveness property encapsulates the concept of a system which will be able to run continuously (not deadlock)
   (a) A Petri Net is considered live if, for all possible markings, there is always a transition enabled.
5. Conservativeness, is associated with the total number of tokens within the Petri Net.
   (a) A Petri Net is said to be strictly conservative if, for all possible markings, the total number of tokens in the Petri Net always remains constant.

9.2.4.1 Examples of Petri Net analysis

1. Figure 9.2 shows a set a simple Petri Nets. As with any Petri Net example, to determine the properties of safeness, boundedness, liveness, and conservativeness one must first construct the reachability tree for each net.
   (a) The initial marking is (1,0) where we can write the marking as the number of token in each place. Thus (1,0) describes a state with 1 token in the first place (P1) and 0 token in the second place (P2). The only enabled transition is T1 - all its input arcs have a token available. If we fire T1, we end up in state (0, 1) - zero tokens in P1, and one token in P2. At this point no further transitions are enabled. We have found all possible states for this Petri Net. The reachability tree showing the states we have found, together with the transitions that must be fired to change between states, is shown in Figure 9.3.
      i. The number of tokens in P1 for all states never exceeds 1. Thus P1 is safe. The same can be said for P2. Since all the places are safe, this net is safe.
      ii. P1 is 1-bounded. P2 is 1-bounded. Thus this net is 1-bounded.
      iii. There is no transition enabled from the second state. Thus this net is not live.
      iv. The total number of tokens in the first state is 1 (1 in P1, 0 in P2). The total number in the second state is 1 (0 in P1, 1 in P2). Since this number remains constant in all states, the net is conservative.
   (b) This net is similar to the previous case, except that there is a transition (T2) enabled in the second state. Firing this transition takes us to state (1,0). Since this state is already in the reachability tree, we just need to add the link. This creates a cycle in the tree.
      i. The number of tokens in P1 for all states never exceeds 1. Thus P1 is safe. The same can be said for P2. Since all the places are safe, this net is safe.
      ii. P1 is 1-bounded. P2 is 1-bounded. Thus this net is 1-bounded.
      iii. In the first state T1 is enabled. In the second state T2 is enabled. Thus the net is live.
      iv. The total number of tokens in the first state is 1 (1 in P1, 0 in P2). The total number in the second state is 1 (0 in P1, 1 in P2). Since this number remains constant in all states, the net is conservative.

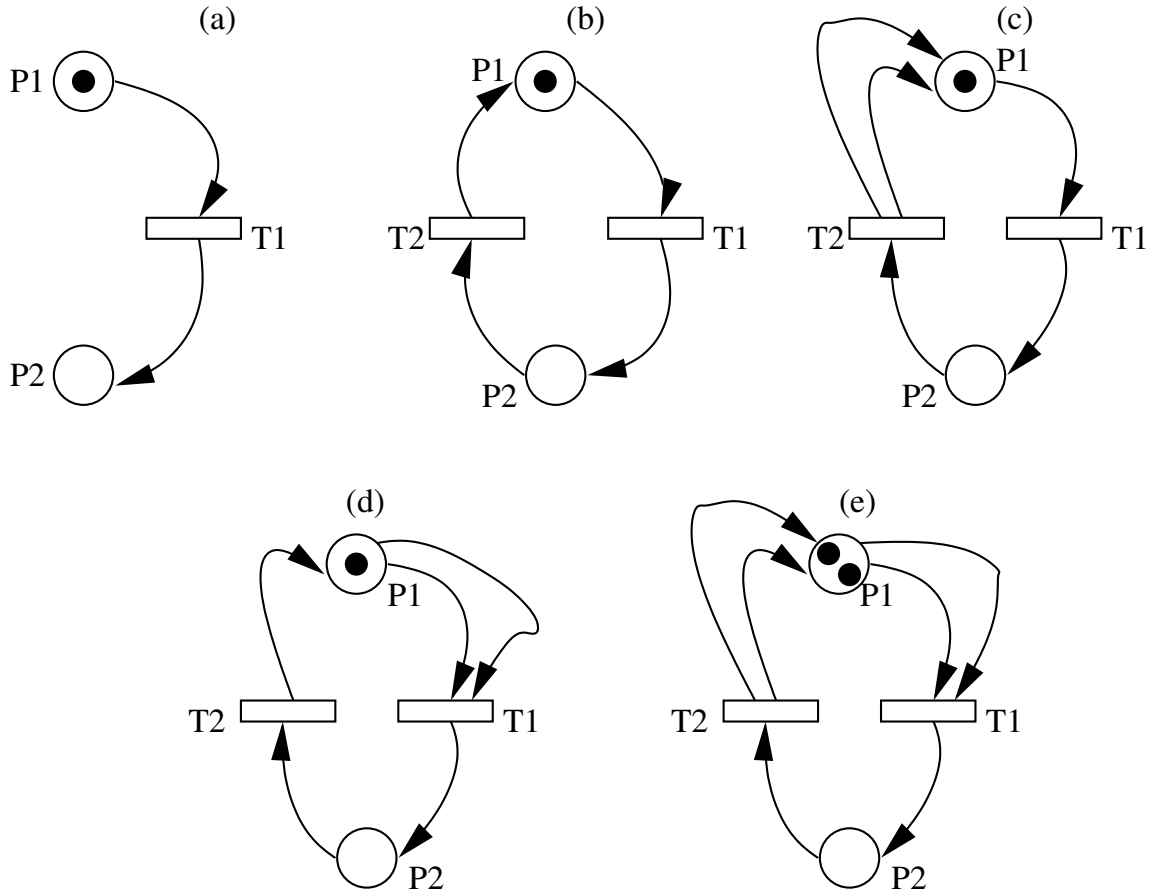


Figure 9.2: Simple Petri Nets

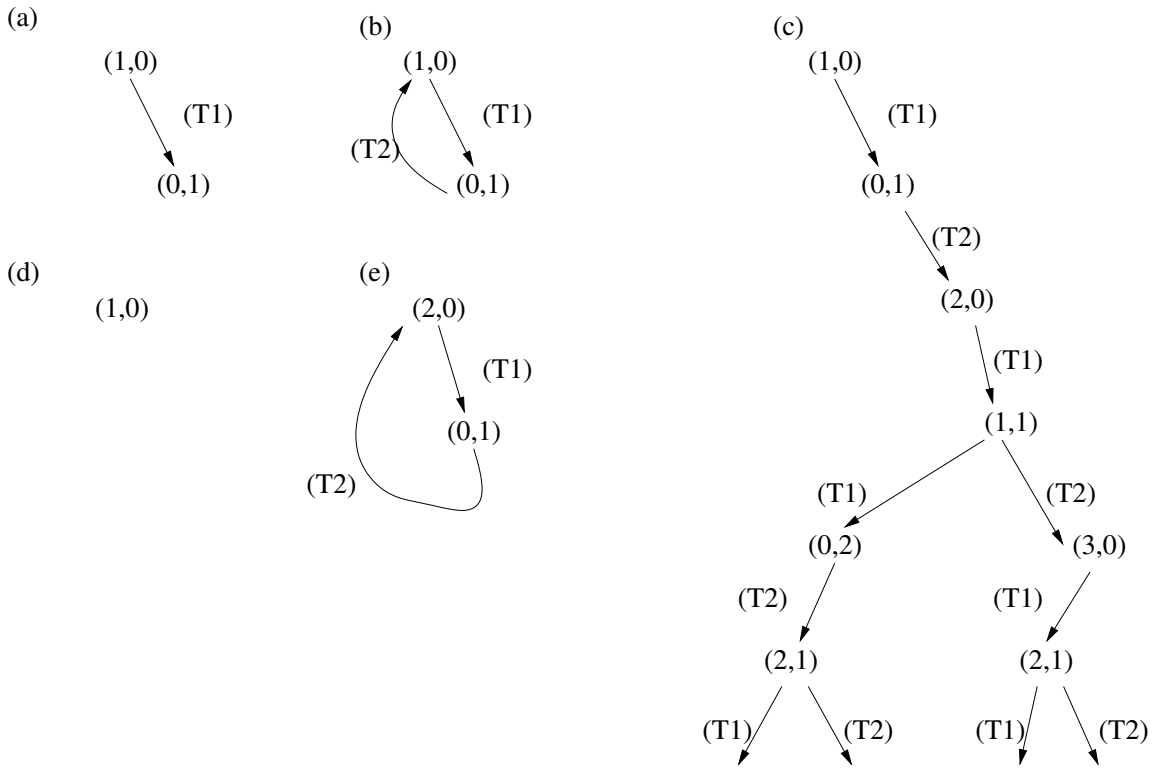


Figure 9.3: Reachability trees for the Simple Petri Nets.

(c) Starting in state (1,0), only T1 is enabled. This fires taking us to state (0,1) with T2 enabled. T2 fires taking to state (2, 0). T1 can fire, taking us to state (1,1). At this point two transitions are enabled. To construct the reachability tree we must try out both cases.

- Firing T1 takes us to state (0,2). T2 is enabled, allowing us to continue to add to the reachability tree at this point.
- Firing T2 takes us to the state (3,1). Both transitions are enabled, creating another branch in the reachability tree.

One can quickly see that the tree is infinite. A portion of the tree is shown in Figure 9.3. One can still make deductions about the properties of the Petri Net.

i. There are places which contain more than one token in some states. Thus the net is not safe.

ii. One can see that the number of tokens in each place can be made as large as desired. Thus none of the places are bounded, and the net as a whole is thus also not bounded.

iii. In any state, at least one of the transitions is enabled. Thus the net is live.

iv. The number of tokens in each state is not constant - consider state (2,0) with 2 tokens, and state (3,1) with 4 tokens. Thus the net is not conservative.

(d) The reachability tree for this case contains a single starting state - (1,0). T1 is not enabled because it requires a token for each of its input links.

i. The number of tokens in P1 for all states never exceeds 1. Thus P1 is safe. The same can be said for P2. Since all the places are safe, this net is safe.

ii. P1 is 1-bounded. P2 is 0-bounded. Thus this net is 1-bounded.

iii. No transitions are enabled in the first state. Thus the net is not live.

iv. The total number of tokens in the first state is 1 (1 in P1, 0 in P2). Since this number remains constant in all states, the net is conservative.

(e) The starting state is (2,0). T1 is enabled, and firing gives state (0,1). T2 is enabled and firing returns the system to state (2,0).

i. The number of tokens in P1 for the first state is 2. This place is not safe. Thus this net is not safe.

ii. P1 is 2-bounded. P2 is 1-bounded. Thus this net is 2-bounded.

iii. All states have transitions enabled. Thus the net is live.

iv. The total number of tokens in the first state is 2 (2 in P1, 0 in P2). The number in the second state is 1 (0 in P1, 1 in P2). Since this number is not constant, the net is not conservative.

9.3 Simulation

Simulation in general is to pretend that one deals with a real thing while really working with an imitation. In operations research the imitation is a computer model of the simulated reality. Also a flight simulator on a PC is a computer model of some aspects of the flight: it shows on the screen the controls and what the "pilot" is supposed to see from the "cockpit".

Simulation of models is often used in industry commerce and military where it is very costly, dangerous or often impossible to make experiments with real systems. Provided that models are adequate descriptions of reality, experimenting with them can save money, suffering and even time.

Simulation are used on systems which change with time such as a gas station where cars come and go (called dynamic systems) and involve randomness (nobody can guess at exactly which time the next car should arrive at the station). Modeling complex dynamic systems theoretically needs too many simplifications and the emerging models may not be therefore valid. Simulation does not require that many simplifying assumptions, making it the only tool even in absence of randomness.

Thus with simulation use a mathematical or logical model, driven by randomly generated event times (inter-arrival and service times, for example) to approximate the evolution of the system over time. We then take averages over this evolution, and hope that they give us insight into the performance of the actual system.

It is a cold hard fact that many times we may simply have no alternative to using simulation to analyze, and also possible try to optimize, a system. One must be particularly cautious though, as simulation can be a subtle thing, and there are many common mistakes which can be made. We will discuss some of these in later sections.

9.3.1 Implementing Simulation

Simulators can be grouped into two categories:

Continuous simulators are characterized by the extensive use of mathematical formulae which describe how a simulated component responds when subjected to various conditions. For example, consider a circuit described at the transistor, resistor and capacitor level. The behaviour of all these components are well understood and are governed by several equations which describe their respective behaviours. A continuous simulator would apply those equations in the context of the components' environment and connectivity and produce a continuous graph which accurately reflects how the components would react if they were actually hooked up in reality. The graphs usually reflect the changes in the state of the system with respect to time; however, other relationships may also be demonstrated as well. Unfortunately, the mathematical equations employed by a continuous simulator can make the simulation very computationally intensive, especially in the presence of thousands of interconnected elements. As such, continuous simulators may be slow and are consequently only useful when simulating a relatively small number of components which are described at a low level of abstraction. Example: simulation of an analogue circuit.

Discrete-event simulation is used to simulate components which normally operate at a higher level of abstraction than components simulated by continuous simulators. Within the context of discrete-event simulation, an event is defined as an incident which causes the system to change its state in some way. For example, a new event is created whenever a simulation component generates output. A succession of these events provide an effective dynamic model of the system being simulated. What separates discrete-event simulation from continuous simulation is the fact that the events in a discrete-event simulator can occur only during a distinct unit of time during the simulation - events are not permitted to occur in between time units. Discrete event simulation is generally more popular than continuous simulation because it is usually faster while also providing a reasonably accurate approximation of a system's behaviour. Example: simulation of a digital circuit.

Monte Carlo simulation is related to discrete-event simulation. Monte Carlo simulators usually make extensive use of random number generators in order to simulate the desired system. Unlike discrete-event simulators, which are often used to model deterministic systems, Monte Carlo simulators can be used to effectively model systems in which probability and nondeterminism plays a major role. As such, Monte Carlo simulators are commonly used to model stochastic systems.

Discussion of simulation will be confined to discrete-event simulation.

9.3.2 Simulation concepts

The following examples introduce some concepts related to simulation.

Suppose we wish to procure some items for a special store promotion (call them pans). We can buy these pans for $22 and sell them for $35. If we have any pans left after the promotion we can sell them to a discounter for $15. If we run out of special pans, we will sell normal pans, of which we have an unlimited supply, and which cost us $32. We must buy the special pans in advance. How many pans should we purchase?

Clearly, the answer depends on the demand. We do not know the demand that the promotion will generate. We do have enough experience in this business to have some feel for the probabilities involved. Suppose the probabilities are:

Demand	Probability
8	0.1
9	0.2
10	0.3
11	0.2
12	0.1
13	0.1

To simulate a possible demand, we will generate a random value between 0 and 1. Suppose I generate the random number .78. How can I generate a demand. Simply assign each demand to a range of values proportional to its probability and determine where the .78 occurs. One possibility is:

Demand	Range
8	0.0 - 0.099
9	0.1 - 0.299
10	0.3 - 0.599
11	0.6 - 0.799
12	0.8 - 0.899
13	0.9 - 0.999

Looking at the ranges, we see the demand is 11. The demand for .35 is 10 while that for .98 is 13.

How can we use this random demand? Suppose we have decided to procure 10 pans. We could determine the total profit for each of our random demands: the profit for the first is 133, for the second is 130, and 139 for the third. Our estimate for the profit if we order 10 is $134.

We could then go on to check the profit if we order a different amount. For instance, if we order 13, our profit is estimated at $162.33.

At the end, we would have a guess at the best order quantity, and an estimate of the expected profit. We would not know for certain that we made the right move, but statistical analysis can estimate how far off we might be (by the way, to get statistical significance for this problem, you need roughly 20 runs at each demand level). Note also, for this problem there is an analytic solution.

A bank is planning on installing an automated teller machine and must choose between buying one Zippy machine or two Klunky machines. A Zippy costs exactly twice one Klunky to buy and operate, so the goal of the bank is simply to provide the best service.

From the data available, it appears that customers arrive according to a Poisson process at the rate of 1 per minute. Zippy provides service that is exponential with mean .9 minutes. Each Klunky provides service that is exponential with mean 1.8 minutes. We will assume that customers lined up for the the two Klunkies will form a single queue. The performance measure we will use is the average time waiting in the queue for the first 100 customers (the bank has decided it is most irritating to wait and customers are pacified if they are being served). Should the bank buy one Zippy or two Klunkies?

One method would be to install one Zippy for a few weeks, measure the average wait, and then rip it out and install two Klunkies and measure their wait. If necessary, then, the Klunkies could be ripped out, and the Zippy reinstalled.

Simulation, of course, gives a much more appealing solution. We can simply create a computer simulation of the above experiment. To do this by hand, we would generate (perhaps by using a table of random numbers) a series of arrival times and service times and determine how long the wait was. For instance, we might end up with arrival times of .2, .7, 1.6, 2.3, 3.4, 5.6, and so on and service times for Zippy of .9, .7, 2.3, 1.6, .1, .6, and so on (and double that amount for Klunkies). The simulation for one Zippy would then have a customer arrive at time .2 and go right to the machine. At time .7 a customer arrives and waits in line. At time 1.1, customer 1 leaves, and customer 2 uses the machine (having waited .4). Customer 3 arrives at 1.6, customer 2 leaves at 1.8, allowing customer 3 (after having waited .2) to use the machine (total wait so far is .6). And so on. Similar analysis could be done with the two Klunky system. Fortunately, we can have a computer do all of the work for us.

9.3.3 Simulation Algorithms

There are three major ways to approach discrete simulation. These are event scheduling, activity scanning, and process orientation. Each approach is offers a different way to look at a simulation problem. Each, in its own way, suggests mechanisms to model real situations.

9.3.3.1 Event scheduling

Event scheduling is the first way simulations were developed. An event is anything that changes the system statistics (also known as the state of the system) other than the mere passage of time. The essential idea of event scheduling is to move along the time scale until an event occurs and then, depending on the event, modify the system state and possibly schedule new events.

In our Zippy versus Klunky example, the events can be simply the arrival of a customer and the finish of the customer. The following summarizes the actions associated with these events:

9.3.3.1.1 Arrival Event

1. Check the status of the ATM(s) (idle or busy)
   (a) If there is an idle machine
      i. Start serving the customer, update idle status
      ii. Generate a departure event for this customer

(b) If all busy, place customer in queue

2. Generate new arrival event for next customer.

9.3.3.1.2 Departure Event

1. Check queue (empty or not)

   (a) If empty, set ATM to idle

   (b) If not empty then do

      i. Choose a waiting customer, start serving the customer

      ii. Generate a departure event for this customer

We will see in the next section how to generate the random times needed in order to be able to generate such things as the service times and the arrival times.

Based on this, it is a trivial exercise to run through a simulation of the system. The events are stored in an event queue, which lists all events in order. The first event in the queue is taken off, and other events may then be added (assuming that an event only triggers other events in the future).

Conventionally, a data structure known as a global event queue is used to process and manage the events and to activate components as required during the simulation.

A sequential simulation algorithm repeatedly performs the following three steps:

```
Remove the event with the minimum time-stamp
  from the input queue using the head operation.
Set the simulated time to the time for this event.
Execute that event possibly generating new events.
Insert newly generated events into the input queue.
```

9.3.3.2 Activity scanning

There is a related, but subtly different, approach called activity scanning. This is perhaps illustrated by our example. In this model, there are three activities:

1. An arrival occurs

2. A service is completed

3. A service begins

The actions associated with these activities are

1. (Arrival) Put customer in queue, generate next arrival

2. (Completion) Declare ATM idle

3. (Begin) Make ATM busy, remove customer from queue, generate completion.

If you compare with the previous subsection, you see that there is one new activity: the beginning of service. It is not triggered by any other activity. Instead, it occurs when the following two conditions are satisfied:

1. There are customers in the queue

2. An ATM is idle

We call such an activity a conditional activity (also called a conditional event). As such, the system must be continually scanned for the satisfaction of the conditions of the activity.

What is the advantage of this approach? Mainly, it is much simpler to think about and formalize. When first asked to generate the events of a simulation, many people come up with events that require activity scanning. The main disadvantage with this approach is inefficiency. It is difficult to get the computer to always scan for such activities without slowing down the simulation.

To date, the disadvantages of such an approach outweigh the advantages. I suspect, however, that due to the relationship of this approach with artificial intelligence and rule based systems, this approach will become increasingly popular.

9.3.3.3 Process oriented modeling

The process oriented approach differs from the previous methods by providing tools for the user to define the system, rather than the details of its operation. In our case, we would view the system as having three essential components:

1. A SOURCE that generates arriving customers
2. A QUEUE to hold waiting customers
3. A FACILITY to serve customers

If a program provides these components as basic building blocks, then the modeler need only provide the parameters: the inter-arrival times, the queue discipline, and the service times.

At this point, the computer must do all of the following work:

```
- SOURCE must generate customers
- When a customer arrives, it must either be placed in
    QUEUE or sent to FACILITY
- When a customer arrives at FACILITY, a processing time
    must be generated
- When FACILITY becomes idle, the QUEUE must be checked
    for more customers.
```

From a modeler point of view, the system looks quite different: events have no meaning. From a processor point, however, the system is the same: events must be scheduled and the system updated.

Given a good set of basic building blocks, this approach can result in very simple, intuitive, believable simulations.

9.3.4 Generating Sample Values

The arrival times for items of data and processing time for various events can be deterministic, allowing the same sequence of events to recur each time the simulation is run. More realistically, these times are regarded as random values, conforming to some statistical distribution. Timing information can be captured from real systems, more often the parameters defining the distribution of real data are determined, and used to shape the distribution of a sequence of random numbers.

A key component of any stochastic simulation is the generation of the event times. The first thing one needs to generate event times is data. You need to have some idea of what the distribution of the event times looks like, or else you have no way of simulating it.

You may have either:

- A general distribution which you believe the times come from (uniform, exponential, normal).
- Discrete data points which you use to approximate the distribution.

In both of the cases, we generate a random number uniformly distributed between zero and one, and transform that into a sampled value from the distribution. This is called a Monte Carlo simulation.

There is no such thing as a truly random computer generated number. Most of the generation routines use a seed, multiplication, addition and modular division to generate the next value from the last one. This means that if the number 0.624 appears in a sequence, followed by the number 0.192, then every time, and in every sequence where 0.624 appears, it will be followed by 0.192.

Does this mean that things are hopeless? No. Some people have developed quite good random (actually properly called pseudo-random) number generation routines. Also, by changing seeds periodically, you can "shake things up". One should bear this in mind though, and understand that this immediately should cause one to take simulation based results with a grain of salt.

9.3.4.1 Discrete Data Points

Assume we are given N data points from the distribution we are attempting to simulate. These cover a range of values from x_1 to x_m. (m and N will likely be different values as we assume we have duplications.) Assume that the value x_i appears d_i times in our sample.

Then we estimate the probability mass function of x_i, p_i as $\frac{d_i}{N}$, as well as the cumulative distribution function of the distribution, $F(i)$. (To calculate $F(i)$, we simply sum the mass functions for values less than or equal to i.)

For example, if we are given $N = 50$ data points for the length of a call, which range between one and five minutes:

i	Number of Occurrences	p_i	$F(i)$
1	6	0.12	0.12
2	23	0.46	0.58
3	19	0.38	0.96
4	0	0.00	.096
5	2	0.04	1

To simulate a number from this distribution, we generate a random number from our computer, r. (Recall that $r \in [0,1]$.) We then compare this with ranges of the cumulative distribution function of the distribution to get our value.

- If $0 \leq r < 0.12$ then our simulated call length equals one.
- If $0.12 \leq r < 0.58$ then our simulated call length equals two.
- If $0.58 \leq r < 0.96$ then our simulated call length equals three.
- If $0.96 \leq r < 0.96$ then our simulated call length equals four. (This can not happen.)
- If $0.96 \leq r < 1$ then our simulated call length equals five.

9.3.4.2 Known Distribution Function

There are two methods for use here.

9.3.4.2.1 Inverse Transformation Method Assume that the inter-arrival time is known to follow a certain distribution, F. Then if we can take the inverse of the distribution function F, we can use this to simulate values from F.

This is achieved as follows.

1. Attain $F(x)$ for the random variable, either from basic knowledge or summing or integrating the density or mass function.

2. Generate a random number r.
3. Set $F(x) = r$, and solve for x. This allows you to determine the inverse distribution function $F^{-1}(x)$, and use this to map r to x.

9.3.4.2.2 Acceptance-Rejection Method If we cannot take the inverse of a distribution, but we have its density function, $f(x)$, and it lies on a finite interval $[a,b]$, then we can use the acceptance-rejection method.

1. Set $M = max(f(x) : x \in [a,b])$.
2. Generate two random numbers, r_1 and r_2.
3. Compute $x^* = a + (b-a)r_1$.
4. If $r_2 \leq \frac{f(x^*)}{M}$ we accept x^* as our value. If not, we reject x^* and repeat the procedure.

This weights the probability of accepting each value by its mass.

9.3.5 Building the Model

Now that we know how to generate random values from a distribution, we can move toward building a simulation model for a system we are interested in. We assume that the system can be in one of many states, and its evolution from state to state is affected by random variables.

We will concern ourselves with discrete state systems, where the system changes state only at discrete time points, when specific events occur. A queue is an example of a discrete state system.

In contrast to this, a continuous state system is one that is constantly changing. The stock market is an example of this. These are more complex to deal with, and often require special and complex models (such as Brownian motion, in the case of the stock market.)

We will also concentrate on dynamic simulation - a system as it evolves over time. Static simulation represents only a particular point or value in time, such as the Monte Carlo methods to generate random samples above.

9.3.5.1 Model Choice

The heart of a good simulation is an accurate model of the system. (See, there is no way to avoid stochastic modeling.) In some sense, the accuracy of the model is constrained by the accuracy of the estimated distributions of the random variables in the system.

A general guideline to use while modeling is to try to keep the model as simple as possible, and then once you understand the system and have developed confidence in the validity of your model, you can add more complicated features.

An important part of deciding upon a model is specifying the states of the system. One seeks a state which contains all of the relevant information, and which is observable for the system.

Modeling is a very delicate process, and in many cases is the most difficult portion of a simulation.

9.3.5.2 Evolution of the System

We must now describe how we control the evolution of the system. To do so we define an event as an occurrence which causes the state of the system to change.

The time between events is kept track of using clocks, and an event list, which tells us what the next scheduled events are and when they will occur. When an event occurs, the systems state is updated, and new clocks are set as necessary for the times between events. These clocks are set with values sampled from the random distributions. When a clock "expires" the event it was keeping time for occurs.

We can either run the simulation until a certain event occurs, or more often for a fixed amount of time, say T time units.

EXAMPLE: Assume a queuing system, with the queue starting empty, at 6:30 am. Then the only event which can occur is a customer arrival. To determine when this will happen we get a value from our inter-arrival distribution, and set this as the value on the arrival clock. Say this time is 40 seconds. We let this amount of time pass (as nothing happens in this interval), and then when this clock "expires" the arrival occurs, at time 6:30:40.

At this time the state of the system changes. We update the number in system to one. This causes a service to be initiated (the customer enters service), so we need a value for this clock. We get this from a service time distribution (assume for now that all of the servers are the same). Say this value is 4 minutes. So this customer will exit the system at 6:34:40. This is added to the event list.

We must also get a new time for the next inter-arrival. Lets say this is 30 seconds. So this will occur at 6:31:10. This is added to the event list.

We then let time run until the minimum of the clocks expires, this time another arrival, at time 6:31:10. We increment the number of customers in the system by one (to two), get a new service time for this customer (say 90 seconds, to complete at 6:32:40), and get a new inter-arrival time (say 2 minutes, 6:33:10). These are added to the event list.

We then let time pass to the next event, which in this case is a service completion for the second customer. This occurs at time 6:32:40. We decrement the number in system by one, and if there were anyone in queue they would enter service, and we would set their clock. As it is, no new clocks need to be set, so we proceed to the next event, which will be at arrival at time 6:33:10.

We proceed this way, always selecting the minimum time remaining on any clock for the next event. This is called next event time-advance simulation. There is also fixed-increment time advance simulation, but this is less widely used.

A few tips:

- It is a good idea to list the actions which must be taken when each particular event occurs; which state variables change, and what clocks must be set. Writing this out in your modeling phase will help ensure you do not forget anything, and will also prove useful when transferring your model to the computer.
  For example, when a service completion occurs, the number in system is decremented by one, and if there is anyone in queue this customer enters service and his clock time must be set. (If we have different servers, we must keep track also of where he enters service, as thus determine what distribution his clock time is sampled from.) If there is no one in queue, this need not be done. It is common to generate a file of random numbers prior to executing the simulation. This not only saves time, but also allows one to use the same simulated values when comparing two different configurations of the same system. This reduces the variation in the comparison, and is in a sense more "fair".

- If all service times were exponential, we could then keep one clock, as the minimum of a set of exponential random variables is a single exponential random variable with a rate which is the sum of the rates of the individual exponentials.

9.3.6 Analyzing the output

Once a model to a system has been created and the input distributions have been determined, it is possible to run the simulation and get data about aspects of the system. For instance, in our ATM model, we were concerned with the average waiting time for the first one hundred customers. Other things we might want to analyze are the average queue length, the maximum queue length, and the total time in system.

There are two fundamentally different types of systems that might be analyzed. The first is called a terminating system. In a terminating system, there is a natural beginning to the system and a natural end to the system. Here we are concerned with questions like "What is the average waiting time of the first 100 customers?" and "What is the maximum queue length in the first hour of operation?" This contrasts with steady state systems which do not have such natural beginnings and endings. The sort of question asked here is "How many customers per hour can this system handle" and "In the long run, what will be the average queue length?" The reason these differ is due to the effect of the transient period: a period

of time at the beginning of a simulation that is not typical of the long run due to the effect of the starting configuration. There is an initial transient period after which the system settles down into its steady state behaviour.

9.3.6.1 Simulation Statistics

The purpose of doing a simulation is to estimate something about the behavior of a system. We must keep these a record of what we are interested in.

For example, if we are interested in the time average number in system, we must keep track of the number in system as part of our state variable. As we are interested in a time average here, we will eventually divide our quantity by T. Note though that we must weight each value by the length of time it persists.

If we do n replications of a system, giving us estimates $x_1, x_2, \ldots, x_n$ for a desired quantity (like delay), then we can take the sample average as our overall estimate of the delay as: $\bar{X} = \sum_{i=1}^{n} \frac{x_i}{n}$

If we are interested in the average delay of a customer, we must keep track of when each customer arrives, and when he departs. This then becomes part of the state of the system, as for each customer in the system we must record when he arrived. (This could become very memory intensive.) We would then sum all of the individual delays, and divide by the total number of customers served in a day, to get the customer average delay.

9.3.6.1.1 Statistical Analysis We use simulation to try to estimate the performance of our system. As the system evolves according to a random process, namely the random event times we generate, different simulations of the same system will yield different estimates of the desired quantity. We need some way of developing an estimate of this quantity from our simulations.

There are two typical problems which can cloud our simulation results.

Autocorrelation: The evolution of a system, a queue, an inventory process, etc., is such that the values along the sample path, or particular evolution the simulation takes, are correlated with one another.
If we were just taking independent samples of different values from a distribution, we could take a long string of them, sum them up and divide to get an estimate of their mean. But what if the values influenced each other? What if the first one being large caused the second one to be more likely to be large, and so on? Then it is possible that the effect of one really large value could last a long time, and influence almost the entire simulation!
A little thought will show you that this indeed can happen when we simulate things like queues. If the queue gets large initially from one very large service time, it may stay large for quite a while, inflating my estimate for the entire sample path. This is autocorrelation.
To combat this, instead of doing one long simulation, we do a series of different replications (n say) of the same system, with different random inputs. This way, as different replications are independent, we can average the output of all n replications with each other, and hope in this way to counteract the effect of one unusually large (and autocorrelated) sample path.

Transience: We might be interested in the steady-state number in system, but when we start the simulation, the system may start empty. This initial emptiness influences the early values of my simulation – in some sense it is unusual, and thus these values cannot be considered as good indicators of the steady-state, which I want.
To combat this the early observations of a system are usually discarded; i.e. not used in the calculation of averages. This may be the first 50, 500, or 5000 data points. The premise is that after a certain amount of time the system settles into steady state, and this is when we want to start counting.
This raises the question of how we know when the system reaches steady-state, so we can start using the values from our simulation? The answer is that we don't. And if you err, it is best to err on the conservative side, of throwing out too many, rather than too few data points.

Note that if we are doing a series of n replications, and discarding m initial values from each replication, we are actually not using nm values from our simulation. This gets "expensive", but there is no simple way around it.

When all is said and done, the major goal of any simulation is to calculate one or more observable properties. This is true whether or not the value of that observable has been determined or can ever be determined through some other means. Good questions to ask yourself whenever reading a computational paper are the following: What observable property is being investigated? Is there some experimental comparison for this property?

The goal of a simulation is now to sample the value of over a representative portion of state space so that a good estimate of the observable can be made. Don't be confused on this point when people discuss how long a simulation ran: What they mean is was the simulation run long enough so that a representative portion of state space was sampled.

9.3.7 The Simulation Process

We now give a general overview of the simulation process.

1. State your objectives. decide what you want to know from your system – what quantities you will try to estimate, what hypothesis you want to test.
2. Build your model. Decide on the model you plan to use. Define your states, what will comprise your event list, and what actions each event will trigger. Estimate the parameters of your model. Start simple.
3. Collect your data. This must be done early, as this may influence what model you use. If you lack data for a process, you will have difficulty modeling it. If it is important enough, you can try to gather additional data before you proceed.
4. Develop the computer implementation. This may involve using a package, or writing the code yourself. If you are doing it yourself, it will also entail random number generation. Advantages of packages are that they are pre-coded, and should be debugged. Disadvantages are that you must try to understand, and trust, someone else's work.
5. Verify the program. Before you proceed to the actual simulation, you should test your code on "toy" models, the answers for which you either know or for which you have a good estimate.
6. Validate the model. Before finally embarking on the simulation, perform one final check to see that your model, and your code, sufficiently describes what you are interested in.
7. Go! If satisfied with all of the above, run the simulations, and calculate your estimates of the results.

One of the advantages of simulation is that at this point, if all goes well, you have a convenient way of experimenting with different system configurations.

Note that comparisons of different configurations should be done with the same stream of random data.

9.3.8 Limitations on Simulation

The problems with simulation, particularly large, complex simulations are as follows:

Validation: How can you determine that a simulation is a correct model of reality? Errors can creep in in many places. The program written may not reflect the model. The models for the random variables may be incorrect. Every statistical test has built in assumptions that may or may not hold.

Fuzziness in output: In an analytical model that we can determine such things as that if the service rate equals the input rate then the queue is unstable (tending toward infinite length). A simulation would not be able to determine such results with such accuracy. The statistical nature of the output makes drawing any firm lines difficult.

Specificity of results: Simulations are generally valid for one real world system. Results that hold for one simulation often do not carry over to other, similar, problems.

Computation time: The amount of time to get statistically significant results is usually grossly underestimated. Simulation without statistical analysis is a waste of CPU cycles.

In general, the rule about using simulation is as follows:

Simulation should be used whenever one or both of the following conditions prevail: the assumptions required by the appropriate analytical model are not sufficiently well satisfied by the real system, or the appropriately formulated model has no analytical solution.

Chapter 10

Network Management and Monitoring

In the late 1970's, computer networks had grown from a simple layout of small, separate networks that were not interconnected to larger networks that were interconnected. These larger networks were called internets and their size grew at an exponential rate. Larger the networks became the more difficult it became to manage (monitor and maintain), and it soon became evident that network management was required.

10.1 Chapter Structure

10.1.1 Outcomes

1. Be able to include network management considerations when designing and building computer networks.

2. Collect and present data relating to the performance of computer networks using professional techniques.

10.1.2 Objectives

This chapter:

1. Describes how management of networks can provide data for managing and monitoring the network.

2. Introduces commonly used network management architectures.

3. Explains the effects of standardization on network management.

10.2 Network Management issues

The International Organization for Standards has defined five key areas of network management. These are

1. fault management

2. configuration management

3. security management

4. performance management

5. accounting management.

10.2.1 Performance Management

The goal of performance management is to measure and make available various aspects of network performance so that internetwork performance can be maintained at an acceptable level. Examples of performance variables that might be provided include network throughput, user response times, and line utilization.

Management entities continually monitor performance variables. When a performance threshold is exceeded, an alert is generated and sent to the network management system. This is a reactive system. When performance becomes unacceptable by virtue of an exceeded user-defined threshold, the system reacts by sending a message. Performance management also permits proactive methods. For example, network simulation can be used to project how network growth will affect performance metrics. Such simulation can effectively alert administrators to impending problems, so that counteractive measures can be taken.

10.2.2 Configuration Management

The goal of configuration management is to monitor network and system configuration information so that the effects on network operation of various versions of hardware and software elements can be tracked and managed. Because all hardware and software elements have operational quirks, flaws, or both that might affect network operation, such information is important to maintaining a smooth-running network.

10.2.3 Accounting Management

The goal of accounting management is to measure network utilization parameters so that individual or group uses of the network can be regulated appropriately. Such regulation minimizes network problems (because network resources can be apportioned out based on resource capacities) and maximizes the fairness of network access across all users.

As with performance management, the first step toward appropriate accounting management is to measure utilization of all important network resources. Analysis of the results provides insight into current usage patterns. Usage quotas can be set at this point. Some correction will be required to reach optimal access practices. From that point on, ongoing measurement of resource use can yield billing information as well as information used to assess continued fair and optimal resource utilization.

10.2.4 Fault Management

The goal of fault management is to detect, log, notify users of, and (to the extent possible) automatically fix network problems in order to keep the network running effectively. Because faults can cause down time or unacceptable network degradation, fault management is perhaps the most widely implemented of the ISO network management elements.

10.2.5 Security Management

The goal of security management is to control access to network resources according to local guidelines so that the network cannot be sabotaged (intentionally or unintentionally) and sensitive information cannot be accessed by those without appropriate authorization. For example, a security management subsystem can monitor users logging on to a network resource, refusing access to those who enter inappropriate access codes.

Security management subsystems work by partitioning network resources into authorized and unauthorized areas. For some users, access to any network resources is inappropriate. Such users are usually company outsiders. For other (internal) network users, access to information originating from a particular department is inappropriate. For example, access to human resource files is inappropriate for most users outside the human resource department.

10.3 Implementing Network Management

Network management may be defined as "the process of controlling a complex data network so as to maximize its efficiency and productivity".

The best way in which a network can be managed is to attain each of the five above areas of network management the most efficiently. To aid in this task, network management protocols are used so that the process of network management is automated (run by computers) as much as possible. Network management is used for functions such as to

- monitor printer queues
- set up addresses for devices
- assign priorities for communication
- install software on the network
- manage databases
- manage power on the network

There are many network management protocols available. The two mainstream protocols however are

- SNMP (Simple Network Management Protocol)
- CMIP (Common Management Information Protocol).

SNMPv1 was the first protocol used. SNMPv2 was designed in 1980's and incorporated many of the features of the original SNMP (which is still in wide use today) as well as a few added features that addressed the original protocol's shortcomings. CMIP also designed in 1980's is better organized and contains many more features than either SNMP v1 or v2. CMIP works under the OSI (Open Systems Interconnection) communication stack. Both protocols have some advantages and disadvantages.

The information the SNMP and CMIP can attain from a network is defined as a MIB (Management Information Base). The MIB is structured like a tree. At the top of the tree is the most general information available about a network. Each branch of the tree then gets more detailed into a specific network area, with the leaves of the tree as specific as the MIB can get. For instance, devices may be a parent in the tree, its children being serial devices and parallel devices. The value of these may be 6 , 2, 4 accordingly; with the numbers corresponding to the number of devices attached (2 serial + 4 parallel = 6 total devices). Each node in the MIB tree is referred to as a variable (hence in the above example, devices, serial devices, and parallel devices are all variables, their values being 6, 2, 4 accordingly). The top of a LAN MIB tree is usually referred to as "internet". There is only one MIB tree defined by ISO. However, part of this tree has sections for vendor-specific extensions. Usually each vendor-specific network has its own MIB that contains its own variable names (for instance, IBM has its own MIB, as does Sun, HP, etc..). Although the variable names may be different, the information contained in each vendor-specific MIB tree is generally the same.

10.4 SNMP - Simple Network Management Protocol

10.4.1 Introduction

Since it was developed in 1988, the Simple Network Management Protocol has become the de facto standard for internetwork management. Because it is a simple solution, requiring little code to implement, vendors can easily build SNMP agents to their products. SNMP is extensible, allowing vendors to easily add network management functions to their existing products. SNMP also separates the management architecture from the architecture of the hardware devices, which broadens the base of multi-vendor support. Perhaps most important, unlike other so-called standards, SNMP is not a mere paper specification, but an implementation that is widely available today.

10.4.2 Network Management Architectures

Network management system contains two primary elements: a manager and agents. The Manager is the console through which the network administrator performs network management functions. Agents are the entities that interface to the actual device being managed. Bridges, Hubs, Routers or network servers are examples of managed devices that contain managed objects. These managed objects might be hardware, configuration parameters, performance statistics, and so on, that directly relate to the current operation of the device in question. These objects are arranged in what is known as a virtual information database ,called a management information base, also called MIB. SNMP allows managers and agents to communicate for the purpose of accessing these objects.

10.4.3 Structure of Management Information

In the Manager/Agent paradigm for network management, managed network objects must be logically accessible. Logical accessibility means that management information must be stored somewhere, and therefore, that the information must be retrievable and modifiable. SNMP actually performs the retrieval and modification. The Structure of Management Information (SMI) which is given in RFC 1155, is based on the OSI SMI given in Draft proposal 2684.

The SMI organizes, names, and describes information so that logical access can occur. The SMI states that each managed object must have a name, a syntax and an encoding. The name, an object identifier(OID), uniquely identifies the object. The syntax defines the data type,such as an integer or a string of octets. The encoding describes how the information associated with the managed objects is serialized for transmission between machines.

Each managed object is given a globally unique name called an object identifier (OID). The object identifier is part of a hierarchical name space that is managed by the ISO. The hierarchical structure is used to guarantee that each name is globally unique. In an object identifier, each level of the hierarchy is identified by a number. All SNMP managed objects start with the number 1.3.6.1.

The syntax used for SNMP is the Abstract Syntax Notation One, ASN.1. The encoding used for SNMP is the Basic Encoding Rules, BER. The names used are object identifiers.

10.4.4 SNMP Protocol

SNMP is based on the manager/agent model. SNMP is referred to as "simple" because the agent requires minimal software. Most of the processing power and the data storage resides on the management system, while a complementary subset of those functions resides in the managed system.

To achieve its goal of being simple, SNMP includes a limited set of management commands and responses. The management system issues Get, GetNext and Set messages to retrieve single or multiple object variables or to establish the value of a single variable. The managed agent sends a Response message to complete the Get, GetNext or Set. The managed agent sends an event notification, called a trap to the management system to identify the occurrence of conditions such as threshold that exceeds a predetermined value. In short there are only five primitive operations:

- get (retrieve operation)
- get next (traversal operation)
- get response (indicative operation)
- set (alter operation)
- trap (asynchronous trap operation)

The NMS periodically requests the status of each device (GetRequest) and each agent responds with the status of its device (GetResponse). Making periodic requests is called polling. Polling reduces the burden on the agent because the NMS decides when polls are needed, and the agent simply responds. Polling also reduces the burden on the network because the polls originate from a single system are at a predictable rate.

The shortcoming of polling is that it does not allow for real-time updates. If a problem occurs on a managed device, the manager does not find out until the agent polled. To handle this, SNMP uses a modified polling system called trap-directed polling.

A trap is an interrupt signaled by a predefined event. When a trap event occurs, the SNMP agent does not wait for the manager to poll, instead it immediately sends information to the manager. Traps allow the agent to inform the manager of unusual events while allowing the manager to maintain control of polling. SNMP traps are sent on UDP port 162. The manager sends polls on port 161 and listens for traps on port 162.

10.4.4.1 Outline of the SNMP protocol

Each SNMP managed object belongs to a community. A community is defined by a community name which is an OctetString with 0 to 255 octets in length. Each SNMP message consists of three components

- version number
- community name
- data - a sequence of PDUs associated with the request

10.4.4.2 Security levels with basic SNMP

Authentication: trivial authentication based on plain text community name exchanged in SNMP messages authentication is based on the assumption that the message is not tampered with or interrogated

Authorization: Once community name is validated then agent or manager checks to see if sending address is permitted or has the rights for the requested operation "View" or "Cut" of the objects together with permitted access rights is then derived for that pair (community name, sending address)

10.4.5 What does SNMP access

SNMP access particular instances of an object. All instances of an object in the MIB reside at the leaf nodes of the MIB tree. SNMP Protocol accesses objects by forming an Object identifier of form x.y where x is the "true" OID for the object in the MIB, and y is a suffix specified by the protocol that uniquely identifies a particular instance (e.g.. when accessing a table). For primitive single instance leaf objects use y=0 for example: sysDescr (OID: 1.3.6.1.2.1.1.1) would be referenced in the SNMP protocol by 1.3.6.1.2.1.1.1.0 (i.e sysDescr.0). For single instance of columnar leaf objects (i.e one instance from a table type of object) use y=I1.I2.I3.... (Ii are table indexes) for example.

An example of some of the nodes in the MIB tree is shown in Table 10.1. The abstract syntax notation, ASN.1 is illustrated below:

```
ObjectDescriptor      OBJECT-TYPE
  SYNTAX  ObjectSyntax
  ACCESS  AccessMode
  STATUS  StatusType
  DESCRIPTION    Description    ::= {ObjectGroup    Entry}
```

The following definitions describe the pieces of the macro:

- ObjectDescriptor: Indicates the textual name assigned to the MIB variable being defined.
- ObjectSyntax: Indicates the abstract syntax for the object type. It must be one of:
  - INTEGER
  - OCTET
  - STRING or DisplayString

	Value
iso	1
org	3
dod	6
internet	1
directory	1
mgmt	2
mib-II	1
system	1
sysDescr	1
sysUpTime	3
interfaces	2
ifTable	2
ifEntry	1
ifInOctets	10
ip	4

Table 10.1: A portion of the MIB tree

  - OBJECT IDENTIFIER
  - NULL
  - Network Address
  - Counter
  - Gauge
  - Time Ticks
  - Opaque
  - See RFC 1155 for definitions of each ObjectSyntax variable.

- AccessMode: Specifies the permissions of the object, which can be either:
  - read-only
  - read-write
  - write-only
  - not-accessible

- StatusType: Specifies the status of the object, which can be either:
  - mandatory
  - optional
  - deprecated
  - obsolete

- Description: Specifies a textual description of the purpose of the MIB variable being defined.
- ObjectGroup: Defines the object group for this MIB variable. The ObjectGroup variable identifies the subtree for the MIB variable.
- Entry: Defines the unique location of the MIB variable in the ObjectGroup variable.

An example of the ASN.1 for the MIB entry for sysDescr is shown below:

```
sysDescr  OBJECT-TYPE
  SYNTAX  DisplayString    (SIZE  (0..255))
  ACCESS  read-only
  STATUS  mandatory
  DESCRIPTION  A textual  description  of the entity.
    This  value  should  include  the full  name and
    version  identification  of system's  hardware
    type,software  operating-system,  and networking
    software.  It is mandatory  that  this  only  contain
    printable  ASCII  characters.
  ::= { system  1 }
```

10.4.6 Underlying communication protocols

SNMP assumes that the communication path is a connectionless communication subnetwork.In other words, no prearranged communication path is established prior to the transmission of data. As a result, SNMP makes no guarantees about the reliable delivery of the data, although in practice most messages get through, and those that don't can be retransmitted. The primary protocols that SNMP implements are the User Datagram Protocol (UDP) and the Internet Protocol (IP). SNMP also requires Data Link Layer protocols such as Ethernet or Token Ring to implement the communication channel from the management to the managed agent.

SNMP's simplicity and connectionless communication also produce a degree of robustness. Neither the manager nor the agent relies on the other for its operation.Thus, a manager may continue to function even if a remote agent fails. When the agent resumes functioning, it can send a trap to the manager, notifying it of its change in operational status. The connectionless nature of SNMP leaves the recovery and error detection up to the NMS (Network Management Station) and even up to the agent. However keep in mind that SNMP is actually transport independent (although original design was connectionless transport function, which corresponds to the UDP protocol) and can be implemented on other transports as well.

10.4.7 The details of the SNMP protocol

This description of SNMP is from Request for Comments: 1157: A Simple Network Management Protocol (SNMP). It defines a simple protocol by which management information for a network element may be inspected or altered by logically remote users.

The Internet Activities Board recommends that all IP and TCP implementations be network manageable. This implies implementation of the Internet MIB (RFC-1156) and at least one of the two recommended management protocols SNMP (RFC-1157) or CMOT (RFC-1095).

10.4.7.1 The SNMP Architecture

Implicit in the SNMP architectural model is a collection of network management stations and network elements. Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, gateways, terminal servers, and the like, which have management agents responsible for performing the network management functions requested by the network management stations. The Simple Network Management Protocol (SNMP) is used to communicate management information between the network management stations and the agents in the network elements.

10.4.7.1.1 Goals of the Architecture The SNMP explicitly minimizes the number and complexity of management functions realized by the management agent itself. This goal is attractive in at least four respects:

1. The development cost for management agent software necessary to support the protocol is accordingly reduced.

2. The degree of management function that is remotely supported is accordingly increased, thereby admitting fullest use of internet resources in the management task.

3. The degree of management function that is remotely supported is accordingly increased, thereby imposing the fewest possible restrictions on the form and sophistication of management tools.

4. Simplified sets of management functions are easily understood and used by developers of network management tools.

A second goal of the protocol is that the functional paradigm for monitoring and control be sufficiently extensible to accommodate additional, possibly unanticipated aspects of network operation and management. A third goal is that the architecture be, as much as possible, independent of the architecture and mechanisms of particular hosts or particular gateways.

10.4.7.1.2 Representation Management information communicated by operation of the SNMP is represented according to the subset of the ASN.1 language that is specified for the definition of non-aggregate types. The SNMP continues and extends this tradition by utilizing a moderately more complex subset of ASN.1 for describing managed objects and for describing the protocol data units used for managing those objects. In addition, the desire to ease eventual transition to OSI-based network management protocols led to the definition in the ASN.1 language of an Internet-standard Structure of Management Information (SMI) and Management Information Base (MIB). The use of the ASN.1 language, was, in part, encouraged by the successful use of ASN.1 in earlier efforts, in particular, the SGMP.

Also for the sake of simplicity, the SNMP uses only a subset of the basic encoding rules of ASN.1. Namely, all encodings use the definite-length form. Further, whenever permissible, non-constructor encodings are used rather than constructor encodings. This restriction applies to all aspects of ASN.1 encoding, both for the top-level protocol data units and the data objects they contain.

10.4.7.1.3 Operations The SNMP models all management agent functions as alterations or inspections of variables. Thus, a protocol entity on a logically remote host (possibly the network element itself) interacts with the management agent resident on the network element in order to retrieve (get) or alter (set) variables. This strategy has at least two positive consequences:

1. It has the effect of limiting the number of essential management functions realized by the management agent to two: one operation to assign a value to a specified configuration or other parameter and another to retrieve such a value.

2. A second effect of this decision is to avoid introducing into the protocol definition support for imperative management commands: the number of such commands is in practice ever-increasing, and the semantics of such commands are in general arbitrarily complex.

The strategy implicit in the SNMP is that the monitoring of network state at any significant level of detail is accomplished primarily by polling for appropriate information on the part of the monitoring center(s). A limited number of unsolicited messages (traps) guide the timing and focus of the polling. Limiting the number of unsolicited messages is consistent with the goal of simplicity and minimizing the amount of traffic generated by the network management function. The exclusion of imperative commands from the set of explicitly supported management functions is unlikely to preclude any desirable management agent operation.

Currently, most commands are requests either to set the value of some parameter or to retrieve such a value, and the function of the few imperative commands currently supported is easily accommodated in an asynchronous mode by this management model. In this scheme, an imperative command might be realized as the setting of a parameter value that subsequently triggers the desired action. For example, rather than implementing a "reboot command," this action might be invoked by simply setting a parameter indicating the number of seconds until system reboot.

10.4.7.1.4 Transport The communication of management information among management entities is realized in the SNMP through the exchange of protocol messages. Consistent with the goal of minimizing complexity of the management agent, the exchange of SNMP messages requires only an unreliable datagram service, and every message is entirely and independently represented by a single transport datagram. While this document specifies the exchange of messages via the UDP protocol, the mechanisms of the SNMP are generally suitable for use with a wide variety of transport services.

10.4.7.1.5 Administrative Relationships The SNMP architecture admits a variety of administrative relationships among entities that participate in the protocol. The entities residing at management stations and network elements which communicate with one another using the SNMP are termed SNMP application entities. The peer processes which implement the SNMP, and thus support the SNMP application entities, are termed protocol entities.

A pairing of an SNMP agent with some arbitrary set of SNMP application entities is called an SNMP community. Each SNMP community is named by a string of octets, that is called the community name for said community.

An SNMP message originated by an SNMP application entity that in fact belongs to the SNMP community named by the community component of said message is called an authentic SNMP message. The set of rules by which an SNMP message is identified as an authentic SNMP message for a particular SNMP community is called an authentication scheme. An implementation of a function that identifies authentic SNMP messages according to one or more authentication schemes is called an authentication service.

Clearly, effective management of administrative relationships among SNMP application entities requires authentication services that (by the use of encryption or other techniques) are able to identify authentic SNMP messages with a high degree of certainty. Some SNMP implementations may wish to support only a trivial authentication service that identifies all SNMP messages as authentic SNMP messages. For any network element, a subset of objects in the MIB that pertain to that element is called a SNMP MIB view.

An element of the set { READ-ONLY, READ-WRITE } is called an SNMP access mode. A pairing of a SNMP access mode with a SNMP MIB view is called an SNMP community profile. A SNMP community profile represents specified access privileges to variables in a specified MIB view. For every variable in the MIB view in a given SNMP community profile, access to that variable is represented by the profile according to the following conventions:

1. if said variable is defined in the MIB with "Access:" of "none," it is unavailable as an operand for any operator;

2. if said variable is defined in the MIB with "Access:" of "read-write" or "write-only" and the access mode of the given profile is READ-WRITE, that variable is available as an operand for the get, set, and trap operations;

3. otherwise, the variable is available as an operand for the get and trap operations.

4. In those cases where a "write-only" variable is an operand used for the get or trap operations, the value given for the variable is implementation-specific.

A pairing of a SNMP community with a SNMP community profile is called a SNMP access policy. An access policy represents a specified community profile afforded by the SNMP agent of a specified SNMP community to other members of that community. All administrative relationships among SNMP application entities are architecturally defined in terms of SNMP access policies. For every SNMP access policy, if the network element on which the SNMP agent for the specified SNMP community resides is not that to which the MIB view for the specified profile pertains, then that policy is called a SNMP proxy access policy. The SNMP agent associated with a proxy access policy is called a SNMP proxy agent.

While careless definition of proxy access policies can result in management loops, prudent definition of proxy policies is useful in at least two ways:

1. It permits the monitoring and control of network elements which are otherwise not addressable using the management protocol and the transport protocol. That is, a proxy agent may provide a protocol

conversion function allowing a management station to apply a consistent management framework to all network elements, including devices such as modems, multiplexers, and other devices which support different management frameworks.

2. It potentially shields network elements from elaborate access control policies. For example, a proxy agent may implement sophisticated access control whereby diverse subsets of variables within the MIB are made accessible to different management stations without increasing the complexity of the network element.

10.4.7.1.6 Naming The names for all object types in the MIB are defined explicitly in the Internet-standard MIB or in other documents. Each instance of any object type defined in the MIB is identified in SNMP operations by a unique name called its "variable name." In general, the name of an SNMP variable is an OBJECT IDENTIFIER of the form x.y, where x is the name of a non-aggregate object type defined in the MIB and y is an OBJECT IDENTIFIER fragment that identifies the desired instance.

This naming strategy admits the fullest exploitation of the semantics of the GetNextRequest-PDU because it assigns names for related variables so as to be contiguous in the lexicographical ordering of all variable names known in the MIB.

The type-specific naming of object instances is defined below for a number of classes of object types. Instances of an object type to which none of the following naming conventions are applicable are named by OBJECT IDENTIFIERs of the form x.0, where x is the name of said object type in the MIB definition.

For example, suppose one wanted to identify an instance of the variable sysDescr. The object class for sysDescr is:

iso	org	dod	internet	mgmt	mib	system	sysDescr
1	3	6	1	2	1	1	1

Hence, the object type, x, would be 1.3.6.1.2.1.1.1 to which is appended an instance sub-identifier of 0. That is, 1.3.6.1.2.1.1.1.0 identifies the one and only instance of sysDescr.

10.4.7.2 Protocol Specification

The network management protocol is an application protocol by which the variables of an agent's MIB may be inspected or altered. Communication among protocol entities is accomplished by the exchange of messages, each of which is entirely and independently represented within a single UDP datagram using the basic encoding rules of ASN.1.

A message consists of a version identifier, an SNMP community name, and a protocol data unit (PDU). A protocol entity receives messages at UDP port 161 on the host with which it is associated for all messages except for those which report traps (i.e., all messages except those which contain the Trap-PDU). Messages which report traps should be received on UDP port 162 for further processing.

An implementation of this protocol need not accept messages whose length exceeds 484 octets. However, it is recommended that implementations support larger datagrams whenever feasible.

It is mandatory that all implementations of the SNMP support the five PDUs:

- GetRequest-PDU,
- GetNextRequest-PDU,
- GetResponse-PDU,
- SetRequest-PDU,
- Trap-PDU.

In the case of the UDP, a transport address consists of an IP address along with a UDP port. Other transport services may be used to support the SNMP. In these cases, the definition of a transport address should be made accordingly.

The top-level actions of a protocol entity which generates a message are as follows:

1. It first constructs the appropriate PDU, e.g., the GetRequest-PDU, as an ASN.1 object.

2. It then passes this ASN.1 object along with a community name its source transport address and the destination transport address, to the service which implements the desired authentication scheme. This authentication service returns another ASN.1 object.

3. The protocol entity then constructs an ASN.1 Message object, using the community name and the resulting ASN.1 object.

4. This new ASN.1 object is then serialized, using the basic encoding rules of ASN.1, and then sent using a transport service to the peer protocol entity.

Similarly, the top-level actions of a protocol entity which receives a message are as follows:

1. It performs a rudimentary parse of the incoming datagram to build an ASN.1 object corresponding to an ASN.1 Message object. If the parse fails, it discards the datagram and performs no further actions.

2. It then verifies the version number of the SNMP message. If there is a mismatch, it discards the datagram and performs no further actions.

3. The protocol entity then passes the community name and user data found in the ASN.1 Message object, along with the datagram's source and destination transport addresses to the service which implements the desired authentication scheme. This entity returns another ASN.1 object, or signals an authentication failure. In the latter case, the protocol entity notes this failure, (possibly) generates a trap, and discards the datagram and performs no further actions.

4. The protocol entity then performs a rudimentary parse on the ASN.1 object returned from the authentication service to build an ASN.1 object corresponding to an ASN.1 PDUs object. If the parse fails, it discards the datagram and performs no further actions. Otherwise, using the named SNMP community, the appropriate profile is selected, and the PDU is processed accordingly. If, as a result of this processing, a message is returned then the source transport address that the response message is sent from shall be identical to the destination transport address that the original request message was sent to.

10.4.7.2.1 The GetRequest-PDU Upon receipt of the GetRequest-PDU, the receiving protocol entity responds according to any applicable rule in the list below:

1. If the object's name does not exactly match the name of some object available or the object is an aggregate type, then error noSuchName is returned.

2. If the size of the GetResponse-PDU generated would exceed a local limitation, then error tooBig is returned.

3. If the value of the object cannot be retrieved, then error genErr is returned.

4. If none of the foregoing rules apply, then the receiving protocol entity sends to the originator of the received message the GetResponse-PDU such that, for each object named in the variable-bindings field of the received message, the corresponding component of the GetResponse-PDU represents the name and value of that variable.

10.4.7.2.2 The GetNextRequest-PDU The receiving protocol entity sends to the originator of the received message the GetResponse-PDU such that, for each name in the variable-bindings field of the received message, the corresponding component of the GetResponse-PDU represents the name and value of that object whose name is, in the lexicographical ordering of the names of all objects available for get operations in the relevant MIB view, together with the value of the name field of the given component, the immediate successor to that value.

Suppose that a routing table has three entries:

Destination	NextHop	Metric
10.0.0.99	89.1.1.42	5
9.1.2.3	99.0.0.3	3
10.0.0.51	89.1.1.42	5

The management station sends to the SNMP agent a GetNextRequest-PDU containing the indicated OBJECT IDENTIFIER values as the requested variable names:

```
GetNextRequest    ( ipRouteDest,
                    ipRouteNextHop,
                    ipRouteMetric1     )
```

The SNMP agent responds with a GetResponse-PDU:

```
GetResponse   (( ipRouteDest.9.1.2.3      = "9.1.2.3"    ),
               ( ipRouteNextHop.9.1.2.     3 = "99.0.0.3"    ),
               ( ipRouteMetric1.9.1.2.     3 = 3 ))
```

The management station continues with:

```
GetNextRequest    ( ipRouteDest.9.1.2.3     ,
                    ipRouteNextHop.9.1.     2.3,
                    ipRouteMetric1.9.1.     2.3 )
```

The SNMP agent responds:

```
GetResponse   (( ipRouteDest.10.0.0.51      = "10.0.0.51"    ),
               ( ipRouteNextHop.10.0.0      .51 = "89.1.1.42"    ),
               ( ipRouteMetric1.10.0.0      .51 = 5 ))
```

The management station continues with:

```
GetNextRequest    ( ipRouteDest.10.0.0.     51,
                    ipRouteNextHop.10.0     .0.51,
                    ipRouteMetric1.10.0     .0.51  )
```

The SNMP agent responds:

```
GetResponse   (( ipRouteDest.10.0.0.99      = "10.0.0.99"    ),
               ( ipRouteNextHop.10.0.0      .99 = "89.1.1.42"    ),
               ( ipRouteMetric1.10.0.0      .99 = 5 ))
```

The management station continues with:

```
GetNextRequest    ( ipRouteDest.10.0.0.     99,
                    ipRouteNextHop.10.0     .0.99,
                    ipRouteMetric1.10.0     .0.99  )
```

As there are no further entries in the table, the SNMP agent returns those objects that are next in the lexicographical ordering of the known object names. This response signals the end of the routing table to the management station. 4.1.4.

10.4.7.2.3 The GetResponse-PDU The form of the GetResponse-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. The GetResponse-PDU is generated by a protocol entity only upon receipt of the GetRequest-PDU, GetNextRequest-PDU, or SetRequest-PDU.

10.4.7.2.4 The SetRequest-PDU The form of the SetRequest-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. For each object named in the variable-bindings field of the received message, the corresponding value is assigned to the variable. Each variable assignment specified by the SetRequest-PDU should be effected as if simultaneously set with respect to all other assignments specified in the same message. The receiving entity then sends to the originator of the received message the GetResponse-PDU of identical form except that the value of the error-status field of the generated message is noError and the value of the error-index field is zero.

10.4.7.2.5 The Trap-PDU The Trap-PDU is generated by a protocol entity only at the request of the SNMP application entity. The means by which an SNMP application entity selects the destination addresses of the SNMP application entities is implementation-specific. Upon receipt of the Trap-PDU, the receiving protocol entity presents its contents to its SNMP application entity.

The significance of the variable-bindings component of the Trap-PDU is implementation-specific. Interpretations of the value of the generic-trap field are:

- The coldStart Trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.
- The warmStart Trap signifies that the sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.
- The linkDown Trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration. The Trap-PDU of type linkDown contains as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.
- The linkUp Trap signifies that the sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up.
- The authenticationFailure Trap signifies that the sending protocol entity is the addressee of a protocol message that is not properly authenticated.

10.4.8 The Advantages of SNMP

The largest advantage to using SNMP is that its design is simple, hence it is easy to implement on a large network, for it neither takes a long time to set up nor poses a lot of stress on the network. Also, its simple design makes it easy for a user to program variables they would like to have monitored, for in a more low-level perspective each variable consists of the following information:

- the variable title
- the data type of the variable (integer, string)
- whether the variable is read-only or read-write
- the value of the variable

The net result of this simplicity is a network manager that is easy to implement and not too stressful on an existing network.

Another advantage of SNMP is that it is in very wide use today. This popularity came about when no other network managers appeared to replace the "band-aid" implementation of SNMP. The result of this is that almost all major vendors of internetwork hardware, such as bridges and routers, design their products to support SNMP, making it very easy to implement.

Expandability is another benefit of SNMP. Because of its simple design, it is easy for the protocol to be updated so that it can expand to the needs of users in the future.

10.4.9 The Disadvantages to SNMP and how they can be Overcome

SNMP is by no means a perfect network manager. It has its faults, yet because of its clever design most of these faults have workarounds.

The first deficiency with SNMP is that it has some large security gaps that can give network intruders access to the information carried along the network. Intruders could also potentially shut down some terminals.

The solution to this problem is simple. Because of the expandability of SNMP, the latest version of SNMP, called SNMPv2, has added some security mechanisms that help combat the largest security problems:

- privacy of data (to prevent intruders from gaining access to information carried along the network),
- authentication (to prevent intruders from sending false data across the network),
- access control (which restricts access of particular variables to certain users, thus removing the possibility of a user accidentally crashing the network).

The biggest problem with SNMP though is that it is generally considered to be so simple that the information it deals with is neither detailed nor well-organized enough to deal with the expanding networks of the 1990's. This is mainly due to the quick creation of SNMP, for it was never intended to lead network management into the 1990's.

This large problem has been fixed in a newer release of SNMP, SNMPv2. This new version allows for more in-detail specification of variables, including the use of the table data structure for easier data retrieval. Also included are two new PDU's that are used to manipulate the tabled objects. In fact, so many new features have been added that the formal specifications for SNMP have expanded from 36 pages (with v1) to 416 pages with SNMPv2. Some may argue that with SNMPv2 the protocol lost its simplicity, but the fact is that changes to SNMP were necessary. It was an old system that just could not handle the network-intensive world of the 1990's. SNMPv2 incorporates a Remote Monitoring MIB (RMON), and enhanced security procedures.

Unfortunately there are at least 4 versions of SNMPv2 which have not been adopted widely. The more significant benefits of SNMPv2 have been incorporated into SNMPv3, described in section 10.6.

10.4.9.1 RMON

The RMON specification defines standard network monitoring functions and interfaces for communicating between SNMP-based devices. RMON gives networks the capability to provide an effective and efficient way to monitor subnetwork-wide behavior while reducing the burden both on other agents and management stations.

The RMON MIB uses an agent device connected to a broadcast network for collecting network traffic statistics. The RMON MIB also performs calculations directly at the agent and does not rely on the manager for all of its functions. Typically, an agent is only responsible for management information that relates to its own device. Without a remote monitoring function, it is difficult, if not impossible, for a manager to construct a profile of any activity on an individual subnetwork.

The RMON MIB can be implemented directly into today's management applications and does not require the entire SNMPv2 to be used. To be effective, a dedicated management station with RMON management and agent capability is attached to the central LAN. The RMON agents are resident on devices that monitor each subnetwork to which they are attached, thereby giving the manager network layer monitoring. This monitoring includes off-line operation, problem detection and reporting, value-added data and multiple-manager support.

10.5 Common Management Information Protocol (CMIP)

The CMIP protocol was supposed to be the protocol that replaced SNMP in the late 1980's. Funded by governments and large corporations, many thought that it would become a reality because of its almost un-

limited development budget. Unfortunately, problems with its implementation have delayed its widespread availability and it is now only available in limited form from its developers themselves.

CMIP was designed to build on SNMP by making up for SNMP's shortcomings and becoming a bigger, more detailed network manager. Its basic design is similar to SNMP, whereby PDU's are employed as variables to monitor a network. CMIP however contains 11 types of PDU's (compared to SNMP's five).

In CMIP, the variables are seen as very complex and sophisticated data structures, with many attributes. These include :

1. variable attributes: which represent the variables characteristics (its data type, whether it is writable).
2. variable behaviors: what actions of that variable can be triggered.
3. Notifications: the variable generates an event report whenever a specified event occurs (such as a terminal shutdown would cause a variable notification event.

As a comparison, SNMP only employs variable properties one and three from above.

10.5.1 The Advantages to the CMIP Approach

The biggest feature of the CMIP protocol is that its variables not only relay information to and from the terminal (as in SNMP), but they can also be used to perform tasks that would be impossible under SNMP. For instance, if a terminal on a network cannot reach its file server a pre-determined amount of times, then CMIP can notify the appropriate personnel of the event. With SNMP however a user would have to explicitly keep track of how many unsuccessful attempts to reach a file server that a terminal has incurred. CMIP thus results in a more efficient network management system, as less work is required by a user to keep updated on the status of their network.

Another advantage to the CMIP approach is that it addresses many of the shortcomings of SNMP. For instance, it has built in security management devices that support authorization, access control, and security logs. The result of this is a safer system from the installation of CMIP; no security upgrades are necessary (like SNMP).

The last advantage is that CMIP was funded by not only governments, but also large corporations. One can induce that CMIP not only has a very large development budget, but also that when it becomes a widely available protocol it will have numerous immediate users, namely the governments and corporations that funded it.

10.5.2 The Disadvantages of CMIP and potential solutions to these problems

From the above, one might wonder that if CMIP is so wonderful, why hasn't it been implemented already (for after all, it has been in development for about ten years). The answer to this is CMIP's only significant disadvantage: the CMIP protocol takes more system resources than SNMP by a factor of ten. In other words, very few systems on this planet would by able to handle a full implementation of CMIP without massive network modifications (such as the installation of thousands of dollars of memory and the purchase of new protocol agents). This major disadvantage has no inexpensive workaround, and for this reason many people believe that the CMIP protocol is doomed to fail. The only possible workaround is to decrease the size of the protocol by changing its specifications. Several protocols have been developed to run "on top" of CMIP and thus use less resources but none of these has gathered enough momentum to challenge SNMP.

Another problem with CMIP is that it is very difficult to program; for the variables require so many different variables that only a few skilled programmers would be able to use the variables to their full potential.

10.6 SNMPv3

Simple Network Management Protocol (SNMP) is the most widely-used network management protocol on TCP/IP-based networks. The functionality of SNMP was enhanced with the publication of SNMPv2.

PDUs (SNMPv1 or SNMPv2)
SNMPv3 Message Headers
UDP
IP

Table 10.2: SNMP architecture.

However, both these versions of SNMP lack security features, notably authentication and privacy, that are required to fully exploit SNMP. A recent set of RFCs, known collectively as SNMPv3, correct this deficiency.

SNMP defines a protocol for the exchange of management information, but does much more than that. It also defines a format for representing management information and a framework for organizing distributing systems into managing systems and managed agents. In addition, a number of specific data base structures, called management information bases (MIBs), have been defined as part of the SNMP suite; these MIBs specify managed objects for the most common network management subjects, including bridges, routers, and LANs. The rapid growth in the popularity of SNMP in the late 1980s and early 1990s led to an awareness of its deficiencies; these fall into the broad categories of functional deficiencies, such as:

- the inability to easily specify the transfer of bulk data
- security deficiencies, such as the lack of authentication and privacy mechanisms.

Many of the functional deficiencies were addressed in a new version of SNMP, known as SNMPv2, first published as a set of RFCs in 1993. The 1993 edition of SNMPv2 also included a security facility, but this was not widely accepted because of a lack of consensus and because of perceived deficiencies in the definition. Accordingly, a revised edition of SNMPv2 was issued in 1996, with the functional enhancements intact but without a security facility. This version used the simple and insecure password-based authentication feature, known as the community feature, provided in SNMPv1, and is referred to as SNMPv2c. To remedy the lack of security, a number of independent groups began work on a security enhancement to SNMPv2. Two competing approaches emerged as front-runners: SNMPv2u and SNMPv2*.

These two approaches served as input to a new IETF SNMPv3 working group. SNMPv3 defines a security capability to be used in conjunction with SNMPv2 (preferred) or SNMPv1, as illustrated in Table 10.2.

Information is exchanged between a management station and an agent in the form of an SNMP message. Security-related processing occurs at the message level. The payload of an SNMP message is either an SNMPv1 or an SNMPv2 protocol data unit (PDU). A PDU indicates a type of management action (e.g., get or set a managed object) and a list of variable names related to that action.

10.6.1 Management principles

The basic idea of any network management system is that there are two types of systems in any networked configuration: agents and managers. Any node in the network that is to be managed, including PCs, workstations, servers, bridges, routers, and so on, includes an agent module. The agent is responsible for

- Collecting and maintaining information about its local environment
- Providing that information to a manager, either in response to a request or in an unsolicited fashion when something noteworthy happens
- Responding to manager commands to alter the local configuration or operating parameters.

The manager station generally provides a user interface so that a human network manager can control and observe the network management process. This interface allows the user to issue commands (e.g.,

deactivate a link, collect statistics on performance, etc.) and provides logic for summarizing and formatting information collected by the system.

Finally, each agent maintains a management information base (MIB) that contains current and historical information about its local configuration and traffic. The management station will maintain a global MIB with summary information from all the agents.

10.6.2 SNMP Architecture

SNMPv3 extends the basic architecture established by the earlier versions, to accommodate alternative security models. It is a modular architecture consisting of a number of SNMP entities. Each SNMP entity includes a single SNMP engine, and a number of applications. It can act as agent, manager or a mixture of the two.

The SNMP engine implements functions for sending and receiving messages, authenticating and encrypting/decrypting messages, and controlling access to managed objects. These functions are provided as services to the applications,

As an examples, in SNMPv3 terminology a traditional SNMP manager includes three categories of applications:

- The Command Generator Applications monitor and manipulate management data at remote agents; they make use of SNMPv1 and/or SNMPv2 PDUs, including Get, GetNext, GetBulk, and Set.
- A Notification Originator Application initiates asynchronous messages; in the case of a traditional manager, the InformRequest PDU is used for this application.
- A Notification Receiver Application processes incoming asynchronous messages; these include InformRequest, SNMPv2-Trap, and SNMPv1 Trap PDUs.

The SNMP engine contains a Dispatcher, a Message Processing Subsystem, and a Security Subsystem. The Dispatcher is a simple traffic manager, and routes each message to the appropriate message processing module. The Message Processing Subsystem reads and writes messages in the appropriate format for the different versions of SNMP. The Security Subsystem performs authentication and encryption functions.

10.6.3 Security in SNMPv3

SNMPv3 provides authentication and privacy services for SNMP. It is designed to secure against the following principal threats:

Modification of Information: An entity could alter an in-transit message generated by an authorized entity in such a way as to effect unauthorized management operations, including the setting of object values. The essence of this threat is that an unauthorized entity could change any management parameter, including those related to configuration, operations, and accounting.

Masquerade: Management operations that are not authorized for some entity may be attempted by that entity by assuming the identity of an authorized entity.

Message Stream Modification: SNMP is designed to operate over a connectionless transport protocol. There is a threat that SNMP messages could be reordered, delayed, or replayed (duplicated) to effect unauthorized management operations. For example, a message to reboot a device could be copied and replayed later.

Disclosure: An entity could observe exchanges between a manager and an agent and thereby learn the values of managed objects and learn of notifiable events. For example, the observation of a set command that changes passwords would enable an attacker to learn the new passwords.

It not intended to secure against the following two threats.

Denial of Service: An attacker may prevent exchanges between a manager and an agent.

Traffic Analysis: An attacker may observe the general pattern of traffic between managers and agents.

Security is accomplished through authentication and encryption. To support these functions, an SNMP engine requires two values: a privacy key (privKey) and an authentication key (authKey). The principles behind the way in which keys are used to achieve the security objectives of SNMPv3 is covered in section 11.6.

Every SNMP agent system in a distributed network has its own unique key for every user authorized to manage it. If multiple users are authorized as managers, the agent has a unique authentication key and a unique encryption key for each user. Thus, if the key for one user is compromised, the keys for other users are not compromised. The keys for a user on different agents are different. Thus, if an agent is compromised, only the user keys for that agent are compromised and not the user keys in use for other agents. Network management can be performed from any point on the network, regardless of the availability of a pre-configured network management system (NMS). This allows a user to perform management functions from any management station. This capability is provided by the password-to-key algorithm described previously.

The SNMPv3 documents define a view-based access control model. This determines whether access to a managed object in a local MIB should be allowed. Various security levels are possible. For example, an agent may allow read-only access for a request communicated in an unauthenticated message but may require authentication for write access. Further, for certain sensitive objects, the agent may require that the request and its response be communicated using the privacy service.

A MIB context is a named subset of the object instances in the local MIB. Contexts provide a useful way of aggregating objects into collections with different access policies. The context is a concept that relates to access control. When a management station interacts with an agent to access management information at the agent, then the interaction is between a management principal and the agent's SNMP engine, and the access control privileges are expressed in a MIB view that applies to this principal and this context.

Chapter 11

Network Security

11.1 Chapter Structure

11.1.1 Outcomes

1. Demonstrate knowledge and understanding of the concepts and principles behind providing security for a computer network.
2. Be able to analyze and critically comment on security issues related to computer networks and networked computers.
3. Be able to discuss issues relevant to working effectively with others situated on remote workstations, using a computer network.
4. Use computer networking technology with due appreciation of the social (particularly security) issues involved.
5. Awareness of resources related to computer security on the Internet.
6. Appreciate that there is a need for life-long learning to remain current in the field of computer network security.

11.1.2 Objectives

This chapter:

1. Describes areas in which security in computer networks can be an issue.
2. Discusses strategies for providing security in computer networks, including some network design case studies.
3. Provides case studies of security breaches and identifies a framework for classification of such breaches.
4. Introduces cryptography as a way for achieving secure communication with remote networked computers.

11.2 Introduction

Most people think of security and cryptography as something used by the military during wars to communicate without the enemy knowing or by governments to keep their secrets.

During the second world war the Germans used their Enigma machines to send messages. They thought those messages were unbreakable since Enigma applied a different code to each letter of a message. However a group of high-IQed British people managed to crack Enigma codes with the world's most advanced computers then, and managed to read Hitler's mail, which helped the Allies winning the war in Europe and Africa. At this same war, on the other side of the globe the United States was braking the codes of Japan with their cryptanalysis team in Pearl Harbor. The Japanese didn't believe their codes could be broken, till the end of the war. The Japanese system was representing each word by a randomly assigned set of five digits. For example, 78934 might stand for "Tokyo", and 78935 for "Suicide". Every time they changed their codes it might be months before the US could read them again. The United States managed to crack about 25% of the Japanese messages, and it was enough.

Julius Caesar, two thousand years earlier, when sending messages used a system of cryptography on his messages to his troops. He used to rotate each letter of the messages by a number of letters, for example, the word ATTACK would become CVVCEM (Rotating each letter 2 letters ahead). Back at those days this system was enough against the semi-literate barbarian spies of the enemy. Today, any 10 year old kid could crack Julius Caesar's messages, after all there are only 26 possibilities to check...

11.3 Security in a Computer Network

Security in a network can come in many different ways. There are basic needs that most networks provide, like Access control, and passwords. Some organizations need better protection of their data, and need more sophisticated means, like the ability to encrypt messages, so that only the receiver will be able to read them (It's quite easy to read messages sent over a normal network). What we may want in our local network :

- Control of access to computers and information.
- Data protection.
- Mail protection.
- Authentication

11.3.1 Control of Access to computers and information

Many operation systems use a password mechanism to control access to the computer. Each user has a login and a password, and whenever he wished to enter the computer, he needs to enter his password. When accessing computer from a terminal (through the network), it's not a good idea to transfer the password as-is, because it is possible to wiretap the network. We might need to encrypt the password, or find a way to use it safely.

When the information is extremely sensitive, we might simply not allow to access it through the network. It is always easier to break through network security than to break into an isolated computer!

11.3.2 Data Protection

Sometimes we need to protect a certain file, so that it won't be available to all. Some operating system provide such protection (In UNIX, a user can decide who can view this file, and who can't, and who can write to it, or execute it), and one could always use a program to encrypt the file. When encrypting the file, we save a scrambled version of it, and then only the ones that are allowed to read the file can decrypt it (un-scramble). The simplest use of encryption needs a key. The encryption program produces a new file, given the original file, and the key. It looks like that:

Encrypted File = Encrypt (Original File, Key).

When we want to open the Encrypted file, we need the Key again :

Original File = Decrypt (Encrypted File, Key)

11.3.3 Mail protection

When we need to send a mail message (or a file through the network, for that matter) we need to be sure that only the intended receiver will be able to read it. Because most network won't guarantee that fact, The messages are usually encrypted. But the encryption scheme described before, is not suitable now. We need to use a key, but we cannot transmit the key to the receiver, because the transmission isn't safe... So we need to know in advance the key that is used, in order to decrypt the message! A better method, is Public Key Encryption. It works like that: Every one has a public key, that is known by all. When we want to send someone an encrypted message, we use his public key. In addition to the public key, everyone also have a Private key, known to himself only. The encrypted message (using the public key) can only be decrypted using the private key! so we could send someone a message, be sure of its safety, without needing to agree upon a key. It works like that :

Encrypted Message = Encrypt (Original Message, Receiver Public Key)

When the receiver gets the message, he opens it, using his Private key :

Original Message = Decrypt (Encrypted Message, Receiver Private Key)

PGP is a high security cryptographic application. PGP allows you to exchange information (whether it's files or messages) with privacy, authentication and convenience. It means that only the intended receiver will be able to access the information and that the sender identity is known. PGP is very easy to use, as there is no need for a dedicated secure channel.

PGP uses two algorithms. One is the Rivest-Shamir-Adleman (RSA) public key cryptosystem and the second is the IDEA algorithm. An IDEA key is generated for the message. The IDEA system is a conventional cryptosystem, so that the same key will be used to decrypt the message. Then RSA is used to encrypt the IDEA key, using the recipient public key. The recipient (When he receives the message) uses his private RSA key to decrypt the IDEA key, which is then used to decrypt the entire message.

11.3.4 Authentication

Another problem with networks, is that we are never sure who sent us a message. It's very easy to write a message pretending to be someone else. A technique called a Digital Signature was developed for that. The sender 'signs' his message, using a key that only he knows. The receiver can then decrypt the signature, just like a regular encrypted message. Again, we usually use a private/public key combination : A signature can only be signed using a private key, and can be decrypted using a public key. In that way, the receiver can be sure as to who sent the message. To ensure authentication and privacy, we can use a digital signature and then encrypt the message. The receiver will need to both decrypt the message using the public key of the sender, and then to authenticate, he'll use his own private key.

11.4 Host Security

11.4.1 The Characteristics of a Secure System and the TCSEC

The Department of Defence Trusted Computer System Evaluation Criteria (TCSEC) also known as the "Orange Book" provides simple but fundamental computer security issues. The table below lists the possible classes of evaluation within the TCSEC.

Class	Requirements
A1	Verified Design
B3	Security Domains
B2	Structured Protection
B1	Labelled Security Protection
C2	Controlled Access Protection
C1	Discretionary Security Protection
D	Minimal Protection

The TCSEC covers trusted systems from Class D (no trust), to class A1 (as trustworthy as the state-of-the-art allows).

Most major operating systems either contain or are easily modifiable to contain the features which are described below. A secure network system has many characteristics with the baseline measurement being the C2-level of security. The C2-level security is defined in the TCSEC and its requirements are :

1. Discretionary Access Control: Discretionary access controls (DAC) are controls which allow a given user to indicate who may or may not view a file that the given user owns. In its simplest form, it allows the user to restrict read, write and execute permissions to specific individuals listed with the access control list (ACL).

2. Object Reuse: Object reuse is the requirement that whenever an object within the Trusted Computing Base (TCB) is no longer required by a given subject, that its contents be erased by the TCB prior to it being returned to free object area. This can be accomplished by writing over the contents a given number of times by a predefined pattern followed by a random pattern. For a system to achieve C2, it must perform this for objects returned in memory as well as objects returned to disk.

3. Identification and Authentication: Identification and authentication is the requirement of the system to require the user to identify his/her self to the TCB. Once the user is identified, the system will request information in order to authenticate the user's identity. The simplest form of I&A is the login ID and password pair.

4. Auditing: Audit is the requirement that the system maintain and protect from modification an audit trail of all access to the object the TCB protects. Access to the audit records will be restricted by the TCB to specific system administrative users.

5. Operational Assurance: This is divided into two aspects, system architecture and system integrity. With regards to system architecture, the system must be able to isolate its resources that are required to be protected so that they can be subject to access control and audit as well as being able to protect itself from external interference or tampering. To attain system integrity, the TCB must contain hardware/software elements which can be used periodically to validate the correct operation of the hardware.

6. Life-Cycle Assurance: Life-cycle assurance is composed of various tests to ensure that the TCB functions as indicated in the documentation and that there are no obvious methods of bypassing or defeating the security mechanisms of the TCB.

7. Documentation

11.4.2 The Security of Windows NT

Windows NT offers features that meet the practical security requirements of business. For every-day users, NT restricts who uses the computer and controls what each authorized user does. For administrators, NT provides tracking and auditing capabilities, enabling network managers to monitor who attempts to use a particular computer and what each user attempts to do.

Networks of computers are becoming increasingly important to most businesses. Networks are used to share key information and resources among many users throughout organizations of various sizes. Frequently, the information stored on network, is secure information that is intended for use only by specific individuals. Therefore, the ability of these networks to prevent unauthorized access to information is paramount to the security and competitiveness of an organization. The NT operating system can maintain the security for these networks.

Within NT, there are two types of configuration, these being a NT server which is shared among a lot of users over a network and the second type of configuration is an NT workstation which is intended for numerous people logging on to this computer. These are used in many Universities and offices. Although they are similar, there are security differences which makes the NT workstation vulnerable to attack.

11.4.2.1 Identification and Authentication

Identification and authentication are the most fundamental security features on an operating system. To log on to NT server/workstation the key combination Ctrl-Alt-Del is pressed. This implements a feature called a "Trusted Path" and this Trusted Path is a requirement of the "Orange Book". This requirement ensures the user that the prompt for username and password is from the operating system and not from a program trying to find out passwords, etc. The user must identify oneself with a username and authenticate oneself with a password.

The sequence for authorizing is as follows :

- User enters the Secure Attention key sequence.
- WinLogon intercepts keystrokes and displays a log-on dialog box.
- WinLogon takes user's input and forwards the text strings to Local Security Authority (LSA).
- LSA sends the strings to the Authentication Package.
- Authentication Package checks for the user in the Accounts Database. Authentication Package returns confirmation or rejection message to LSA.
- LSA notifies WinLogon of authentication result.
- If the user is confirmed, WinLogon passes control to the Win32 subsystem.

Every user belongs to one or more groups and a few special groups are built in. Each group has a name and a set of user rights. Users have the rights of all the groups they belong to, plus any special rights that they are given perhaps from the system-administration.

11.4.2.2 Auditing

Because no system is absolutely secure, administrators need to be able to determine if their system has been the target of attack, or has been vulnerable to the misadventures of a non-malicious user. For NT, the auditing policy is set and controlled with the User Manager. The User Manager provides an easy interface to specify the level of auditing. Because the auditing process contributes to the system overhead, the amount of audit information to be captured has to be carefully weighted in consideration to the overall requirements.

NT divides audited user actions into several categories including file and object access, logging on and off and exercise of user rights. Actions within each category can be audited for success, failure or both.

Audit records consist of three different logs: system events, application events and security events. Each event record is time-stamped and both the process and the user attempting the operation are identified. A log is an object like other resources controlled by NT and therefore it has its own access control list associated with it.

11.4.2.3 Object Reuse

Underlying all of NT's objects are physical RAM and disk space, both of which are continually being recycled for new processes and files. Object Reuse is a security requirement that prevents a user from accessing the remains of another user's work particularly when the operating system creates new objects from previously used resources.

When NT creates a new object for a user , the object is initially empty. For file objects, NT prevents the user from reading past a file's logical end-of-file marker and thus possibly peeking at data from an erased file. Also if a user has a right to extend a file, NT overwrites that area on disk before granting access to it.

When a program allocates memory, NT first clears a section of RAM that a newly created memory object will occupy. This prevents a user from probing random locations in RAM, searching for the vestiges of documents or file buffers that might contain confidential information.

11.4.2.4 Internet Security Issues

The initial release of NT had some gaping security holes which would allow anyone on the Internet to easily delete any of the files on your computer to which that Web server had delete access. Not only that, but their actions wouldn't even have been logged.

NT supports several types of networking:

- Standard TCP/IP services (such as FTP, WWW, SMTP).
- SMB services (native Windows file/print sharing, named pipes, NT RPC).

If you attach such a workstation to the Internet, anyone can connect to any shared directories on that machine, login as Guest and wreck havoc with the file system of that computer. Or they can connect to the registry on that machine (which is always shared, as described below) and mess it up.

NT installs by default with *Everyone* given write access to much of the registry. In NT 3.51, this was a major problem due to the remote registry access feature of the Registry Editor. Any user could manipulate the registry on any server or workstation on which this user has an account, or on which the guest account is enabled. NT 4.0 fixed this problem.

This key is present by default on NT 4.0 server. It is NOT present on NT 4.0 workstation, but can be added. The presence of this key disables remote registry access, other than to administrators. Another way to prevent remote registry access is to remove the permission for Everyone at the root of the HKEY_LOCAL_MACHINE hive. This is the appropriate way to protect the registry for NT 3.51.

NT programs use remote procedure calls (RPCs) to allow various system services to be performed on a remote computer. For example, the ability to modify the registry on remote computers is implemented using remote procedure calls. There are mechanisms in NT for the RPC server to learn the username of the RPC client and then to limit the functions it will perform based on that username.

When using the FTP server that came with NT 3.51, the home directory you specify for the FTP service is only the initial current directory. Ftp users can change their current directory. So if you specify a home directory of c:\ftp, any ftp user can change to c:\ and thence change to any sub-directories under c:\. Normal NTFS permissions will apply, of course, to whatever account the ftp user is running under. If you don't want ftp users to be able to see the root directory of your primary partition, you should create a separate partition for ftp and then configure ftp so that it can only read and/or write to that partition. The IIS FTP server in NT 4.0 does not have this problem.

11.4.3 The Security of UNIX

The UNIX operating system, although now in widespread use in environments concerned about security, was not really designed with security in mind. This does not mean that UNIX does not provide any security mechanisms; indeed, several very good ones are available. However, most "out of the box" installation procedures from companies such as Sun Microsystems still install the operating system in much the same way as it was installed 15 years ago, with little or no security enabled.

UNIX was originally designed by programmers for use by other programmers. The environment in which it was used was one of open co-operation, not one of privacy. Programmers typically collaborated with each other on projects, and hence preferred to be able to share their files with each other without having to climb over security hurdles. Because the first sites outside of Bell Laboratories to install UNIX were university research laboratories, where a similar environment existed, no real need for greater security was seen until some time later.

In the early 1980s, many universities began to move their UNIX systems out of the research laboratories and into the computer centers, allowing (or forcing) the user population as a whole to use this new and wonderful system. Many businesses and government sites began to install UNIX systems as well, particularly as desktop workstations became more powerful and affordable. Thus, the UNIX operating system is no longer being used only in environments where open collaboration is the goal.

To complicate matters, new features have been added to UNIX over the years, making security even more difficult to control. Perhaps the most problematic features are those relating to networking : remote login, remote command execution, network file systems, disk-less workstations, and electronic mail. All

of these features have increased the utility and usability of UNIX by untold amounts. However, these same features, along with the widespread connection of UNIX systems to the Internet and other networks, have opened up many new areas of vulnerability to unauthorized abuse of the system.

Like most multi-user operating systems, UNIX offers a wide variety of security measures. However, because of its roots as a research platform and its development both inside AT&T Bell Laboratories and academia, UNIX has become synonymous with insecure security. To make UNIX compatible with the C2-level of security, modifications and enhancements have to be integrated into the UNIX. The following presents the security on a typical UNIX operating system.

11.4.3.1 Passwords

Under the UNIX OS, all passwords are usually stored in the /etc/passwd file. These passwords are encrypted with a modified version of the Data Encryption Standard (DES), so that it becomes a one way crypt. When a user logs in, the password entered to the OS at the password prompt is encrypted using the modified DES routine and compared against the encrypted password file inside the /etc/passwd file. If the two match, the user is allowed to log into the system.

However, to reduce the effectiveness of key search, a defence called "password salting" is used. This method employs the use of a random12-bit number, the salt, which is appended onto the password when the password is first entered into the system. The string, the 12-bit number and password, is encrypted and stored in the password file. When a user attempts to log in, the salt stored in the password file, is appended to the supplied password, encrypted and compared to the encrypted string. If the strings match, the user is allowed access to the OS. The use of salts increases the complexity of breaking any given password by 212.

To help ensure the quality of passwords chosen, many UNIX OS's,

- refuse passwords less than a minimum length,
- allow both upper and lower case characters, and
- allow almost all special characters.

In most UNIX systems, the user passwords are stored encrypted in the /etc/passwd file. To attain C2 would require the use of a "shadow" file to store the encrypted passwords as well as any password aging information.

11.4.3.2 Discretionary Access Control

All UNIX systems contain a form of Discretionary Access Control (DAC). These controls are of the form :

```
-rwxrwxrwx  user  group  filename
```

where - is one of d (directory), c (character), s (special), b (block), - (a regular file) and r, w, x are read, write and execute respectively.

Each file has a filename, an owner and an associated group. In conjunction with the standard access controls available on UNIX, there is a feature called umask which allows the user to define a default file and directory creation mode.

The National Computer Security Centre (NCSC) has stated that the access controls that currently exist within UNIX are sufficient for the C2-level. Guidelines for higher levels of classification are available and reflect requirements of the B3 class.

11.4.3.3 Object Reuse

Whenever an object that the system manipulates is returned to a common pool of similar objects, the returned object must have all of its internal information removed. To do this, one must successively write binary patterns over the object followed by a random pattern.

11.4.3.4 Auditing

UNIX audits many things. Common audit items include :

- the last time the user logged in;
- commands that a user executed during a login session;
- which users attempted to gain access to root privilege, and
- which users are logged in at any given time.

Most of this data is available to the normal user. Last login time is normally printed at login so the user can see if this is correct. A discrepancy may force the user to change his/her password. The audit records concerning the root are never accessible to anyone but the system administrator.

To ensure an appropriate level of trust in the access controls, audit considerations must be addressed. The controls may require an increased level of protection from attack. This requirement is relative to the existing controls of the UNIX system. Most UNIX systems currently perform some level of audit. However, to achieve a C2 rating, a few additions are required. To achieve a C2 level of trust, the following changes must occur.

- All logins and logouts must be audited in an audit file. Failed login attempts must be recorded also as the port at which entry was attempted.
- All modifications to the user work area must be recorded and the modifier listed in the records.
- The root must be audited, but in an audit file separate from the standard audit record.
- All other relevant security events must be audited and logged in a separate audit record of attempted security breaches.
- C2 requires separation of the audit records.
- Also, all audit records must be timestamped, indicate which user was operating at that time, the event the user was audited for, and success or failure.

11.4.3.5 Networking

Although a system which allows a network can not be considered "trusted" in the "Orange Book" sense, most modern UNIX OS's are indeed networked. The most common network configuration on a UNIX OS is the NFS system pioneered by Sun. This system allows for transparent connection of a wide assortment of hardware. Also NFS allows a user to login in from one machine to another via a utility called rlogin without having to specify a password. This feature is known as trusted users/trusted hosts. This configuration is the reason why the Internet Worm of November 1988 spread so quickly. This problem was eliminated by making the user login every time he/she wished to connect to another computer.

UNIX to UNIX Copy (uucp) is a means of transferring files between two electronic machines. uucp allows for e-mail and for the transferring of other information.

FTP is a file transfer protocol which allows for a special type of file transfer. This is known as "anonymous ftp". If this is set up correctly, then users will be only allowed access to specific files. However, if it is set up with a few bugs, users may have access to the whole file system. To maintain the tightest security, ftp capability should be removed.

The Network File System (NFS) is designed to allow several hosts to share files over the network. One of the most common uses of NFS is to allow disk-less workstations to be installed in offices, while keeping all disk storage in a central location. Distributed by Sun, NFS has no security features enabled. This means that any host on the Internet may access your files via NFS, regardless of whether you trust them or not. Fortunately, there are several easy ways to make NFS more secure.

Normally, NFS translates the super-user id to a special id called "nobody" in order to prevent a user with "root" on a remote workstation from accessing other people's files. This is good for security, but sometimes a nuisance for system administration, since you cannot make changes to files as "root'" through NFS.

11.4.4 Security in Practice

The following are an extract of CERT (formerly Computer Emergency Response Team) advisories, describing a variety of ways in which security of systems have been compromised.

11.4.4.1 W32/Frethem Malicious Code

The CERT/CC has received a number of reports of malicious code known as W32/Frethem. It affects systems running Microsoft Windows with unpatched versions of Internet Explorer and mail clients that use IE's HTML rendering engine (including Outlook and Outlook Express). Patched systems (or systems that do not use IE's HTML rendering engine for mail) may also be affected if a user manually executes the malicious code. A number of variants of this code have been identified.

W32/Frethem arrives as an email message containing three MIME parts with the subject "Re: Your password!" The body of the message is contained in the first MIME part and includes a specially crafted IFRAME tag that will cause the malicious attachment to be executed when this part is rendered in a vulnerable mail user agent.

When run, it gathers the current user's default SMTP server, email address, and display name from the registry keys. It uses these in conjunction with its built-in SMTP engine in order to propagate. It harvests email addresses from the Windows Address Book as well as any other files with .wab, .dbx, .mbx, .mdb, and .eml extensions.

As with other malicious code having mass-mailing capabilities, W32/Frethem may cause denial-of-service conditions in networks where either (a) multiple systems are infected, or (b) large volumes of infected mail are received.

Users are encouraged to install the patches, run and maintain an anti-virus product and exercise caution when receiving email with attachments.

Sites can use email filtering techniques to delete messages containing subject lines known to contain the malicious code, or they can filter all attachments.

11.4.4.2 Exploitation of Vulnerabilities in Microsoft SQL Server

The CERT/CC has received reports of systems being compromised through the automated exploitation of null or weak default sa passwords in Microsoft SQL Server and Microsoft Data Engine. This activity is accompanied by high volumes of scanning, and appears to be related to recently discovered self-propagating malicious code, referred to by various sources as Spida, SQLsnake, and Digispid.

The Spida worm scans for systems listening on port 1433/tcp. Once connected, it attempts to use the xp_cmdshell utility to enable and set a password for the guest user.

If successful, the worm then

1. assigns the guest user to the local Administrator and Domain Admins groups
2. copies itself to the victim system
3. disables the guest account
4. sets the sa password to the same password as the guest account
5. executes the copy on the victim system

Once the local copy is executing on the victim system, the worm begins scanning for other systems to infect. It also attempts to send a copy of the local password (SAM) database, network configuration information, and other SQL server configuration information to a fixed email address (ixtld@postone.com) via email.

The scanning activity of the Spida worm may cause denial-of-service conditions on compromised systems, and it has been reported to cause high traffic volumes even on networks with no compromised hosts. Information about the victim system's configuration and accounts may be compromised by the email the worm attempts to send.

By leveraging a default null password, an attacker may execute arbitrary commands on the system in the security context in which the Microsoft SQL Server services are running. While site-specific configurations may vary, the SQL Server is typically run with system-level privileges.

11.4.4.3 Social Engineering Attacks via IRC and Instant Messaging

The CERT/CC has received reports of social engineering attacks on users of Internet Relay Chat (IRC) and Instant Messaging (IM) services. Intruders trick unsuspecting users into downloading and executing malicious software, which allows the intruders to use the systems as attack platforms for launching distributed denial-of-service (DDoS) attacks. The reports to the CERT/CC indicate that tens of thousands of systems have recently been compromised in this manner.

This is purely a social engineering attack since the user's decision to download and run the software is the deciding factor in whether or not the attack is successful. Although this activity is not novel, the technique is still effective, as evidenced by reports of tens of thousands of systems being compromised in this manner.

As with any DDoS tool installation, the impact is twofold. First, on systems that are compromised by users running untrusted software, intruders may

- exercise remote control
- expose confidential data
- install other malicious software
- change files
- delete files

The malicious code being distributed in these attacks is under continuous development by intruders, but most anti-virus software vendors release frequently updated information, tools, or virus databases to help detect and recover from the malicious code involved in this activity.

Users of IRC and IM services should be particularly wary of following links or running software sent to them by other users, as this is a commonly used method among intruders attempting to build networks of DDoS agents.

11.4.4.4 W32/Gibe Malicious Code

The CERT/CC has received numerous reports of a piece of malicious code, written for the Windows platform, commonly known as W32/Gibe. W32/Gibe spreads via email disguised as a Microsoft security bulletin and patch. A user must execute the attached file in order to be infected. The payload is non-destructive, but a back-door is installed that may allow an intruder access to the system.

W32/Gibe is a Windows binary executable written in Visual Basic that is spreading via email. The email appears to be from Microsoft; however, Microsoft does not distribute patches via email.

When the attached file containing the malicious code is executed, it appears as though it is installing a Microsoft Security Update. It displays several dialog boxes during this process. The malicious code continues to execute regardless of the user's responses to the displayed dialog boxes. (Clicking "Cancel" will not stop the malicious code from executing.)

W32/Gibe installs a back-door (GFXacc.exe), which listens on port 12378/tcp. This may allow an intruder to gain access to the system and execute arbitrary commands. In addition, W32/Gibe mass-mails copies of itself to addresses found on the victim host. The victim and targeted sites may experience an increased load on the mail server when the malicious code is propagating.

11.4.4.5 Exploitation of vulnerability in SSH1 CRC-32 compensation attack detector

The CERT/CC has received multiple reports of systems being compromised via a CRC-32 compensation attack detector vulnerability. We are also receiving reports of increased scanning activity for the SSH service (22/tcp).

There is a remote integer overflow vulnerability in several implementations of the SSH1 protocol. This vulnerability is located in a segment of code that was introduced to defend against exploitation of CRC32 weaknesses in the SSH1 protocol. The attack detection function makes use of a dynamically allocated hash

table to store connection information that is then examined to detect and respond to CRC32 attacks. By sending a crafted SSH1 packet to an affected host, an attacker can cause the SSH daemon to create a hash table with a size of zero. When the detection function then attempts to hash values into the null-sized hash table, these values can be used to modify the return address of the function call, thus causing the program to execute arbitrary code with the privileges of the SSH daemon, typically root.

In reports received by the CERT/CC, systems compromised via this vulnerability have exhibited the following pattern in system log messages:

```
hostname sshd[xxx]: Disconnecting: Corrupted check bytes on input.
hostname sshd[xxx]: Disconnecting: crc32 compensation attack:
network attack detected
hostname sshd[xxx]: Disconnecting: crc32 compensation attack:
network attack detected ...
```

The exploit for this vulnerability appears to use a brute force method, so many messages of this type may be logged before a system is successfully compromised.

The following artifacts have been discovered on systems that were successfully compromised:

- Installation of rootkits that modify standard system utilities to hide the intruder's actions
- Installation of Trojan horse versions of the SSH software, compiled from the latest OpenSSH source code plus intruder-supplied modifications
- Installation of tools to scan large network blocks for other systems that are vulnerable to compromise. Log files left behind from these tools indicate that they operate by looking for the banner displayed upon connection to the sshd service.

An intruder can execute arbitrary code with the privileges of the SSH daemon, typically root.

Solutions include:

- Patch or upgrade to the latest version of the appropriate secure shell software package.
- Disable SSHv1 fall back support. Because the vulnerability affects software handling the SSHv1 protocol, sites may wish to enable SSHv2 support only and disable SSHv1 fallback support. Disabling SSHv1 support is generally a good practice, since a number of other vulnerabilities exist in the SSHv1 protocol itself and software handling of this protocol.
- Restrict access to the secure shell service, to hosts within the network perimeter.

11.4.4.6 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service

The CERT/CC has received reports of new self-propagating malicious code exploiting the vulnerability described in CERT Advisory CA-2001-13 Buffer Overflow In IIS Indexing Service DLL. These reports indicate that the "Code Red" worm has already affected more than 13,000 hosts.

The "Code Red" worm attack sequence proceeds as follows:

- The victim host is scanned for TCP port 80.
- The attacking host sends the exploit string to the victim.
- The worm, now executing on the victim host, checks for the existence of c:\notworm. If found, the worm ceases execution.
- If c:\notworm is not found, the worm begins spawning threads to scan random IP addresses for hosts listening on TCP port 80, exploiting any vulnerable hosts it finds.
- If the victim host's default language is English, then after 100 scanning threads have started and a certain period of time has elapsed following infection, all web pages served by the victim host are defaced.

Each instance of the "Code Red" worm uses the same random number generator seed to create the list of IP addresses it scans. Therefore, each victim host begins scanning the same IP addresses that previous instances have scanned, which could result in a denial of service against the IP addresses earliest in the list.

11.4.4.7 "Carko" Distributed Denial-of-Service Tool

The CERT/CC has received reports that a distributed denial-of-service (DDoS) tool named Carko is being installed on compromised hosts. Based on reports to the CERT/CC, intruders are using the snmpXdmid vulnerability described in the following document to compromise hosts and then install Carko.

Compromised hosts are at high risk for being used to attack other Internet sites, having system binaries and configuration files altered, and exposing sensitive information to external parties. Additionally, DDoS tools are capable of diminishing the availability of services through packet flooding attacks and other resource consumption based attacks.

11.4.4.8 Exploitation of BIND Vulnerabilities

On January 29, 2001 the CERT/CC published CERT Advisory CA-2001-02 detailing multiple vulnerabilities in multiple versions of ISC BIND name-server software. Two of the vulnerabilities described in the advisory are now actively being exploited by the intruder community to compromise systems.

In particular, these vulnerabilities are being exploited:

- VU#325431 - Queries to ISC BIND servers may disclose environment variables
- VU#196945 - ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code

Multiple exploits exist for multiple operating system platforms, and we have seen several versions of packaged kits containing exploits used by intruders to automate the process of scanning for and compromising vulnerable systems. At least one known toolkit employs worm-like techniques designed to cause the attack cycle to self-initiate on a compromised host, which can result in the attack propagating across multiple hosts and networks without intruder interaction. To date, reports to the CERT/CC indicate that successful exploitation has involved hosts running Linux.

11.4.4.9 Exploitation of Hidden File Extensions

There have been a number of recent malicious programs exploiting the default behavior of Windows operating systems to hide file extensions from the user. This behavior can be used to trick users into executing malicious code by making a file appear to be something it is not.

Multiple email-borne viruses are known to exploit the fact that Microsoft Windows operating systems hide certain file extensions. The first major attack incorporating an element of file extension obfuscation was the VBS/LoveLetter worm which contained an email attachment named "LOVE-LETTER-FOR-YOU.TXT.vbs". Other malicious programs have since incorporated similar naming schemes.

The files attached to the email messages sent by these viruses may appear to be harmless text (.txt), MPEG (.mpg), AVI (.avi) or other file types when in fact the file is a malicious script or executable. Users can be tricked into opening a file that appears to be something it is not. A file that appears to be innocent based on it's viewable file name may contain malicious executable code.

11.4.4.10 Buffer Overflow in CDE ToolTalk

The Common Desktop Environment (CDE) ToolTalk RPC database server contains a buffer overflow vulnerability that could allow a remote attacker to execute arbitrary code or cause a denial of service.

The Common Desktop Environment (CDE) is an integrated graphical user interface that runs on UNIX and Linux operating systems. CDE ToolTalk is a message brokering system that provides an architecture for applications to communicate with each other across hosts and platforms. The ToolTalk RPC database server, rpc.ttdbserverd, manages communication between ToolTalk applications. The CDE ToolTalk database server is vulnerable to a heap buffer overflow via an argument passed to the procedure _TT_CREATE_FILE().

An attacker with access to the ToolTalk RPC database service could exploit this vulnerability with a specially crafted RPC message.

Using an RPC message containing a specially crafted argument to _TT_CREATE_FILE(), a remote attacker could execute arbitrary code or cause a denial of service. The ToolTalk database server process runs with root privileges on most systems. Note that the non-executable stack protection provided by some operating systems will not prevent the execution of code located on the heap.

Solution: Apply a patch from your vendor

11.4.4.11 Trojan Horse OpenSSH Distribution

The CERT/CC has received confirmation that some copies of the source code for the OpenSSH package were modified by an intruder and contain a Trojan horse. We strongly encourage sites which employ, redistribute, or mirror the OpenSSH package to immediately verify the integrity of their distribution.

These files appear to have been placed on the FTP server which hosts ftp.openssh.com and ftp.openbsd.org on the 30th or 31st of July, 2002. The OpenSSH development team replaced the Trojan horse copies with the original, un-compromised versions at 13:00 UTC, August 1st, 2002. The Trojan horse copy of the source code was available long enough for copies to propagate to sites that mirror the OpenSSH site.

The Trojan horse versions of OpenSSH contain malicious code that is run when the software is compiled. This code connects to a fixed remote server on 6667/tcp. It can then open a shell running as the user who compiled OpenSSH.

An intruder operating from (or able to impersonate) the remote address specified in the malicious code can gain unauthorized remote access to any host which compiled a version of OpenSSH from this Trojan horse version of the source code. The level of access would be that of the user who compiled the source code.

We encourage sites who downloaded a copy of the OpenSSH distribution to verify the authenticity of their distribution, regardless of where it was obtained. Furthermore, we encourage users to inspect any and all software that may have been downloaded from the compromised site. Note that it is not sufficient to rely on the timestamps or sizes of the file when trying to determine whether or not you have a copy of the Trojan horse version.

You can use the following MD5 checksums to verify the integrity of your OpenSSH source code distribution:

```
459c1d0262e939d6432f1     93c 7a4 ba8 a8  openssh-3.4p1.tar.gz
d5a95626328 7e7fd26152    8db 196 2f2 4c  openssh-3.4p1.tar.gz     .sig
39659226ff5b0d16d0290     b21 f67 c46 f2  openssh-3.4.tgz
9d3e1e31e8d6cdbfa3036     cd1 83a a4a 01  openssh-3.2.2p1.tar.     gz
be4f9ed8da1735efcd770d    c8f a2b b80 8a  openssh-3.2.2p1.tar.     gz.sig
```

Additionally, distributions of the portable release of OpenSSH are distributed with detached PGP signatures. Note that the Trojan horse versions were not signed correctly, and attempts to verify the signatures would have failed.

As a matter of good security practice, the CERT/CC encourages users to verify, whenever possible, the integrity of downloaded software.

11.4.4.12 Vulnerability in PHP

A vulnerability has been discovered in PHP. This vulnerability could be used by a remote attacker to execute arbitrary code or crash PHP and/or the web server. PHP is a popular scripting language in widespread use. The vulnerability occurs in the portion of PHP code responsible for handling file uploads, specifically multipart/form-data. By sending a specially crafted POST request to the web server, an attacker can corrupt the internal data structures used by PHP. Specifically, an intruder can cause an improperly initialized memory structure to be freed. In most cases, an intruder can use this flaw to crash PHP or the web server. Under some circumstances, an intruder may be able to take advantage of this flaw to execute arbitrary code with the privileges of the web server.

You may be aware that freeing memory at inappropriate times in some implementations of malloc and free does not usually result in the execution of arbitrary code. However, because PHP utilizes its own memory management system, the implementation of malloc and free is irrelevant to this problem.

Until patches or an update can be applied, you may wish to deny POST requests, or disable PHP. As a best practice, the CERT/CC recommends disabling all services that are not explicitly required.

11.5 Firewalls

The need for firewalls no longer seems to be in question today. As the Internet and internal corporate networks continue to grow, such a safeguard has become all but mandatory. As a result, network administrators increasingly need to know how to effectively design a firewall. This article explains the basic components and major architectures used in constructing firewalls.

The "right solution" to building a firewall is seldom a single technique; it's usually a carefully crafted combination of techniques to solve different problems. Which problems you need to solve depend on what services you want to provide your users and what level of risk you're willing to accept. Which techniques you use to solve those problems depend on how much time, money, and expertise you have available.

Some protocols (such as Telnet and SMTP) lend themselves to packet filtering. Others (such as., FTP, Archie, Gopher, and WWW) are more effectively handled with proxies. Most firewalls use a combination of proxying and packet filtering.

Before we explore various firewall architectures, let's discuss two major approaches used to build firewalls today: packet filtering and proxy services.

11.5.1 Packet filtering

Packet filtering systems route packets between internal and external hosts, but they do it selectively. They allow or block certain types of packets in a way that reflects a site's own security policy.

Every packet has a set of headers containing certain information. The main information is:

- IP source address
- IP destination address
- Protocol (whether the packet is a TCP, UDP, or ICMP packet)
- TCP or UDP source port
- TCP or UDP destination port
- ICMP message type

In addition, the router knows things about the packet that aren't reflected in the packet headers, such as:

- The interface the packet arrives on
- The interface the packet will go out on

The fact that servers for particular Internet services reside at certain port numbers lets the router block or allow certain types of connections simply by specifying the appropriate port number (such as TCP port 23 for Telnet connections) in the set of rules specified for packet filtering.

Here are some examples of ways in which you might program a screening router to selectively route packets to or from your site:

- Block all incoming connections from systems outside the internal network, except for incoming SMTP connections (so that you can receive email).
- Block all connections to or from certain systems you distrust.

- Allow email and FTP services, but block dangerous services like TFTP, the X Window System, RPC, and the "r" services (rlogin, rsh, rcp, etc.).

To understand how packet filtering works, let's look at the difference between an ordinary router and a screening router.

An ordinary router simply looks at the destination address of each packet and picks the best way it knows to send that packet toward that destination. The decision about how to handle the packet is based solely on its destination. There are two possibilities: the router knows how to send the packet toward its destination, and it does so; or the router does not know how to send the packet toward its destination, and it returns the packet, via an ICMP "destination unreachable" message, to its source.

A screening router, on the other hand, looks at packets more closely. In addition to determining whether or not it can route a packet toward its destination, a screening router also determines whether or not it should. "Should" or "should not" are determined by the site's security policy, which the screening router has been configured to enforce.

Although it is possible for only a screening router to sit between an internal network and the Internet, as shown in the diagram above, this places an enormous responsibility on the screening router. Not only does it need to perform all routing and routing decision-making, but it is the only protecting system; if its security fails (or crumbles under attack), the internal network is exposed. Furthermore, a straightforward screening router can't modify services. A screening router can permit or deny a service, but it can't protect individual operations within a service. If a desirable service has insecure operations, or if the service is normally provided with an insecure server, packet filtering alone can't protect it.

A number of other architectures have evolved to provide additional security in packet filtering firewall implementations.

11.5.2 Proxy services

Proxy services are specialized application or server programs that run on a firewall host: either a dual-homed host with an interface on the internal network and one on the external network, or some other bastion host that has access to the Internet and is accessible from the internal machines. These programs take users' requests for Internet services (such as FTP and Telnet) and forward them, as appropriate according to the site's security policy, to the actual services. The proxies provide replacement connections and act as gateways to the services. For this reason, proxies are sometimes known as application-level gateways.

Proxy services sit, more or less transparently, between a user on the inside (on the internal network) and a service on the outside (on the Internet). Instead of talking to each other directly, each talks to a proxy. Proxies handle all the communication between users and Internet services behind the scenes.

Transparency is the major benefit of proxy services. It's essentially smoke and mirrors. To the user, a proxy server presents the illusion that the user is dealing directly with the real server. To the real server, the proxy server presents the illusion that the real server is dealing directly with a user on the proxy host (as opposed to the user's real host).

Note: Proxy services are effective only when they're used in conjunction with a mechanism that restricts direct communications between the internal and external hosts. Dual-homed hosts and packet filtering are two such mechanisms. If internal hosts are able to communicate directly with external hosts, there's no need for users to use proxy services, and so (in general) they won't.

The proxy server doesn't always just forward users' requests on to the real Internet services. The proxy server can control what users do, because it can make decisions about the requests it processes. Depending on your site's security policy, requests might be allowed or refused. For example, the FTP proxy might refuse to let users export files, or it might allow users to import files only from certain sites. More sophisticated proxy services might allow different capabilities to different hosts, rather than enforcing the same restrictions on all hosts.

11.5.3 Firewall architectures

There are a variety of ways to put various firewalls components together. Let's examine some of these approaches in detail.

11.5.3.1 Dual-homed host architecture

A dual-homed host architecture is built around the dual-homed host computer, a computer that has at least two network interfaces. Such a host could act as a router between the networks these interfaces are attached to; it is capable of routing IP packets from one network to another. However, to implement a dual-homed host type of firewalls architecture, you disable this routing function. Thus, IP packets from one network (such as the Internet) are not directly routed to the other network (such as the internal, protected network). Systems inside the firewall can communicate with the dual-homed host, and systems outside the firewall (on the Internet) can communicate with the dual-homed host, but these systems can't communicate directly with each other. IP traffic between them is completely blocked.

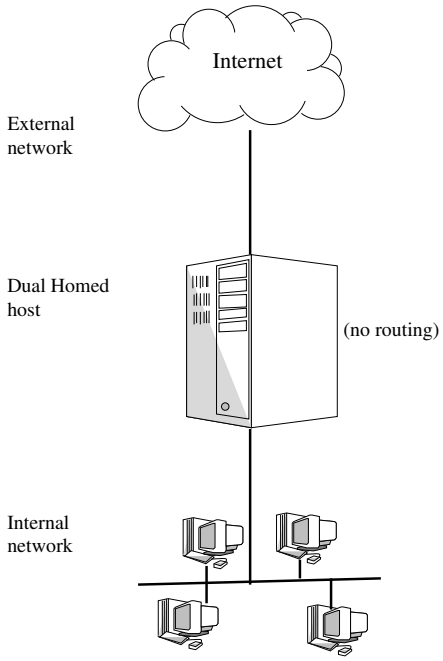


Figure 11.1: Dual-homed host architecture

The network architecture for a dual-homed host firewall is pretty simple: (see Figure 11.1) The dual homed host sits between, and is connected to, the Internet and the internal network. Dual-homed hosts can provide a very high level of control. If you aren't allowing packets to go between external and internal networks at all, you can be sure that any packet on the internal network that has an external source is evidence of some kind of security problem. In some cases, a dual-homed host will allow you to reject connections that claim to be for a particular service but that don't actually contain the right kind of data. (A packet filtering system, on the other hand, has difficulty with this level of control.) However, it takes considerable work to consistently take advantage of the potential advantages of dual-homed hosts.

A dual-homed host can provide services only by proxying them, or by having users log into the dual-homed host directly. User accounts present significant security problems by themselves. They present special problems on dual-homed hosts, where they may unexpectedly enable services you consider insecure. Furthermore, most users find it inconvenient to use a dual-homed host by logging into it.

Proxying is much less problematic, but may not be available for all services you're interested in.

11.5.3.2 Screened host architecture

Whereas a dual-homed host architecture provides services from a host that's attached to multiple networks (but has routing turned off), a screened host architecture provides services from a host that's attached to only the internal network, using a separate router. In this architecture, the primary security is provided by

packet filtering. (For example, packet filtering is what prevents people from going around proxy servers to make direct connections.)

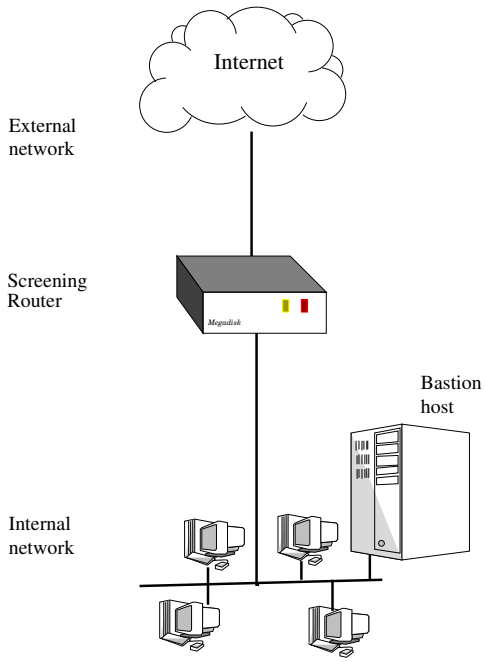


Figure 11.2: Screened host architecture

The bastion host sits on the internal network as in Figure 11.2. The packet filtering on the screening router is set up in such a way that the bastion host is the only system on the internal network that hosts on the Internet can open connections to (for example, to deliver incoming email). Even then, only certain types of connections are allowed. Any external system trying to access internal systems or services will have to connect to this host. The bastion host thus needs to maintain a high level of host security.

The packet filtering also permits the bastion host to open allowable connections to the outside world. The packet filtering configuration in the screening router may do one of the following:

- Allow other internal hosts to open connections to hosts on the Internet for certain services (allowing those services via packet filtering),
- Disallow all connections from internal hosts (forcing those hosts to use proxy services via the bastion host).

You can mix and match these approaches for different services; some may be allowed directly via packet filtering, while others may be allowed only indirectly via proxy. It all depends on the particular policy your site is trying to enforce.

Because this architecture allows packets to move from the Internet to the internal networks, it may seem more risky than a dual-homed host architecture, which is designed so that no external packet can reach the internal network. In practice, however, the dual-homed host architecture is also prone to failures that let packets actually cross from the external network to the internal network. (Because this type of failure is completely unexpected, there are unlikely to be protections against attacks of this kind.) Furthermore, it's easier to defend a router, which provides a very limited set of services, than it is to defend a host. For most purposes, the screened host architecture provides both better security and better usability than the dual-homed host architecture.

Compared to other architectures, however, such as the screened subnet architecture discussed in the following section, there are some disadvantages to the screened host architecture. The major one is that

if an attacker manages to break in to the bastion host, there is nothing left in the way of network security between the bastion host and the rest of the internal hosts. The router also presents a single point of failure; if the router is compromised, the entire network is available to an attacker. For this reason, the screened subnet architecture has become increasingly popular.

11.5.3.3 Screened subnet architecture

The screened subnet architecture adds an extra layer of security to the screened host architecture by adding a perimeter network that further isolates the internal network from the Internet, as in Figure 11.3.

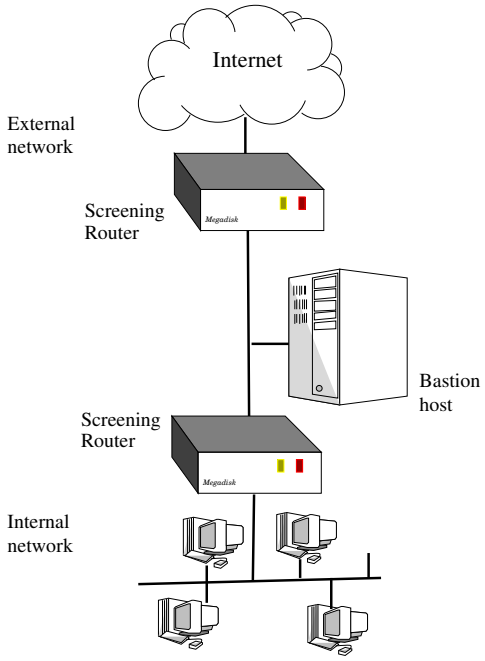


Figure 11.3: Screened subnet architecture

Why do this? By their nature, bastion hosts are the most vulnerable machines on your network. Despite your best efforts to protect them, they are the machines most likely to be attacked, because they're the machines that can be attacked. If, as in a screened host architecture, your internal network is wide open to attack from your bastion host, then your bastion host is a very tempting target. There are no other defenses between it and your other internal machines (besides whatever host security they may have, which is usually very little). If someone successfully breaks into the bastion host in a screened host architecture, he's hit the jackpot.

By isolating the bastion host on a perimeter network, you can reduce the impact of a break-in on the bastion host. It is no longer an instantaneous jackpot; it gives an intruder some access, but not all.

With the simplest type of screened subnet architecture, there are two screening routers, each connected to the perimeter net. One sits between the perimeter net and the internal network, and the other sits between the perimeter net and the external network (usually the Internet). To break into the internal network with this type of architecture, an attacker would have to get past both routers. Even if the attacker somehow broke in to the bastion host, he'd still have to get past the interior router. There is no single vulnerable point that will compromise the internal network.

Some sites go so far as to create a layered series of perimeter nets between the outside world and their interior network. Less trusted and more vulnerable services are placed on the outer perimeter nets, farthest from the interior network. The idea is that an attacker who breaks into a machine on an outer perimeter net will have a harder time successfully attacking internal machines because of the additional layers of security between the outer perimeter and the internal network. This is only true if there is actually some meaning

to the different layers, however; if the filtering systems between each layer allow the same things between all layers, the additional layers don't provide any additional security.

The perimeter network is another layer of security, an additional network between the external network and your protected internal network. If an attacker successfully breaks into the outer reaches of your firewall, the perimeter net offers an additional layer of protection between that attacker and your internal systems.

Here's an example of why a perimeter network can be helpful. In many network setups, it's possible for any machine on a given network to see the traffic for every machine on that network. This is true for most Ethernet-based networks, (and Ethernet is by far the most common local area networking technology in use today); it is also true for several other popular technologies, such as token ring and FDDI. Snoopers may succeed in picking up passwords by watching for those used during Telnet, FTP, and rlogin sessions. Even if passwords aren't compromised, snoopers can still peek at the contents of sensitive files people may be accessing, interesting email they may be reading, and so on; the snooper can essentially "watch over the shoulder" of anyone using the network.

With a perimeter network, if someone breaks into a bastion host on the perimeter net, he'll be able to snoop only on traffic on that net. All the traffic on the perimeter net should be either to or from the bastion host, or to or from the Internet. Because no strictly internal traffic (that is, traffic between two internal hosts, which is presumably sensitive or proprietary) passes over the perimeter net, internal traffic will be safe from prying eyes if the bastion host is compromised.

Obviously, traffic to and from the bastion host, or the external world, will still be visible.

With the screened subnet architecture, you attach a bastion host (or hosts) to the perimeter net; this host is the main point of contact for incoming connections from the outside world; for example:

- For incoming email (SMTP) sessions to deliver electronic mail to the site
- For incoming FTP connections to the site's anonymous FTP server
- For incoming domain name service (DNS) queries about the site

and so on.

Outbound services (from internal clients to servers on the Internet) are handled in either of these ways:

- Set up packet filtering on both the exterior and interior routers to allow internal clients to access external servers directly.
- Set up proxy servers to run on the bastion host (if your firewall uses proxy software) to allow internal clients to access external servers indirectly. You would also set up packet filtering to allow the internal clients to talk to the proxy servers on the bastion host and vice versa, but to prohibit direct communications between internal clients and the outside world.

The interior router (sometimes called the choke router in firewalls literature) protects the internal network from both the Internet and the perimeter net.

The interior router does most of the packet filtering for your firewall. It allows selected services outbound from the internal net to the Internet. These services are the services your site can safely support and safely provide using packet filtering rather than proxies. (Your site needs to establish its own definition of what "safe" means. You'll have to consider your own needs, capabilities, and constraints; there is no one answer for all sites.) The services you allow might include outgoing Telnet, FTP, WAIS, Archie, Gopher, and others, as appropriate for your own needs and concerns.

The services the interior router allows between your bastion host (on the perimeter net itself) and your internal net are not necessarily the same services the interior router allows between the Internet and your internal net. The reason for limiting the services between the bastion host and the internal network is to reduce the number of machines (and the number of services on those machines) that can be attacked from the bastion host, should it be compromised.

You should limit the services allowed between the bastion host and the internal net to just those that are actually needed, such as SMTP (so the bastion host can forward incoming email), DNS (so the bastion host can answer questions from internal machines, or ask them, depending on your configuration), and so

on. You should further limit services, to the extent possible, by allowing them only to or from particular internal hosts; for example, SMTP might be limited only to connections between the bastion host and your internal mail server or servers. Pay careful attention to the security of those remaining internal hosts and services that can be contacted by the bastion host, because those hosts and services will be what an attacker goes after–indeed, will be all the attacker can go after–if the attacker manages to break in to your bastion host.

In theory, the exterior router (sometimes called the access router in firewalls literature) protects both the perimeter net and the internal net from the Internet. In practice, exterior routers tend to allow almost anything outbound from the perimeter net, and they generally do very little packet filtering. The packet filtering rules to protect internal machines would need to be essentially the same on both the interior router and the exterior router; if there's an error in the rules that allows access to an attacker, the error will probably be present on both routers.

Frequently, the exterior router is provided by an external group (for example, your Internet provider), and your access to it may be limited. An external group that's maintaining a router will probably be willing to put in a few general packet filtering rules, but won't want to maintain a complicated or frequently changing rule set. You also may not trust them as much as you trust your own routers. If the router breaks and they install a new one, are they going to remember to reinstall the filters? Are they even going to bother to mention that they replaced the router so that you know to check?

The only packet filtering rules that are really special on the exterior router are those that protect the machines on the perimeter net (that is, the bastion hosts and the internal router). Generally, however, not much protection is necessary, because the hosts on the perimeter net are protected primarily through host security (although redundancy never hurts).

The rest of the rules that you could put on the exterior router are duplicates of the rules on the interior router. These are the rules that prevent insecure traffic from going between internal hosts and the Internet. To support proxy services, where the interior router will let the internal hosts send some protocols as long as they are talking to the bastion host, the exterior router could let those protocols through as long as they are coming from the bastion host. These rules are desirable for an extra level of security, but they're theoretically blocking only packets that can't exist because they've already been blocked by the interior router. If they do exist, either the interior router has failed, or somebody has connected an unexpected host to the perimeter network.

So, what does the exterior router actually need to do? One of the security tasks that the exterior router can usefully perform–a task that usually can't easily be done anywhere else–is the blocking of any incoming packets from the Internet that have forged source addresses. Such packets claim to have come from within the internal network, but actually are coming in from the Internet.

The interior router could do this, but it can't tell if packets that claim to be from the perimeter net are forged. While the perimeter net shouldn't have anything fully trusted on it, it's still going to be more trusted than the external universe; being able to forge packets from it will give an attacker most of the benefits of compromising the bastion host. The exterior router is at a clearer boundary. The interior router also can't protect the systems on the perimeter net against forged packets.

11.5.4 Variations on firewall architectures

However, there is a lot of variation in architectures. There is a good deal of flexibility in how you can configure and combine firewall components to best suit your hardware, your budget, and your security policy.

11.5.4.1 Internal firewalls

In some situations, you may also be protecting parts of your internal network from other parts. There are a number of reasons why you might want to do this:

- You have test or lab networks with strange things going on there.
- You have networks that are less secure than the rest of your site, such as demonstration or teaching networks where outsiders are commonly present.

- You have networks that are more secure than the rest of your site, such as secret development project networks or networks where financial data or grades are passed around.

Laboratory and test networks are often the first networks that people consider separating from the rest of an organization via a firewall (usually as the result of some horrible experience where something escapes the laboratory and runs amok). Unless people are working on routers, this type of firewall can be quite simple. Neither a perimeter net nor a bastion host is needed, because there is no worry about snooping (all users are internal anyway), and you don't need to provide many services (the machines are not people's home machines). In most cases, you'll want a packet filtering router that allows any connection inbound to the test network, but only known safe connections from it. In a few cases (for example, if you are testing bandwidth on the network), you may want to protect the test network from outside traffic that would invalidate tests, in which case you'll deny inbound connections and allow outbound connections.

If you are testing routers, it's probably wisest to use an entirely disconnected network; if you don't do this, then at least prevent the firewall router from listening to routing updates from the test network.

11.5.4.2 Joint venture firewalls

Sometimes, organizations come together for certain limited reasons, such as a joint project; they need to be able to share machines, data, and other resources for the duration of the project. For example, look at the decision of IBM and Apple to collaborate on the PowerPC, a personal computer that runs a common operating system; undertaking one joint project doesn't mean that IBM and Apple have decided to merge their organizations or to open up all their operations to each other.

Although the two parties have decided to trust each other for the purposes of this project, they are still competitors. They want to protect most of their systems and information from each other. It isn't just that they may distrust each other; it's also that they can't be sure how good the other's security is. They don't want to risk that an intruder into their partner's system might, through this joint venture, find a route into their system as well. This security problem occurs even if the collaborators aren't also competitors.

Shared perimeter networks are a good way to approach joint networks. Each party can install its own router, under its own control, onto a perimeter net between the two organizations.

11.6 Cryptography

People mean different things when they talk about cryptography. Children play with toy ciphers and secret languages. However, these have nothing to do with real security and strong encryption. Strong encryption is the kind of encryption that can be used to protect information of real value against organized criminals, multinational corporations, and major governments. Strong encryption used to be only military business; however, in the information society it has become one of the central tools for maintaining privacy and confidentiality.

As we move into an information society, the technological means for global surveillance of millions of individual people are becoming available to major governments. Cryptography has become one of the main tools for privacy, trust, access control, electronic payments, corporate security, and countless other fields.

Cryptography is no longer a military thing that should not be messed with. It is time to demystify cryptography and make full use of the advantages it provides for the modern society. Widespread cryptography is also one of the few defenses people have against suddenly finding themselves in a totalitarian surveillance society that can monitor and control everything they do.

11.6.1 Basic Terminology

Suppose that someone wants to send a message to a receiver, and wants to be sure that no-one else can read the message. However, there is the possibility that someone else opens the letter or hears the electronic communication.

In cryptographic terminology, the message is called plaintext or cleartext. Encoding the contents of the message in such a way that hides its contents from outsiders is called encryption. The encrypted message

is called the ciphertext. The process of retrieving the plaintext from the ciphertext is called decryption. Encryption and decryption usually make use of a key, and the coding method is such that decryption can be performed only by knowing the proper key.

Cryptography is the art or science of keeping messages secret. Cryptanalysis is the art of breaking ciphers, i.e. retrieving the plaintext without knowing the proper key. People who do cryptography are cryptographers, and practitioners of cryptanalysis are cryptanalysts.

Cryptography deals with all aspects of secure messaging, authentication, digital signatures, electronic money, and other applications. Cryptology is the branch of mathematics that studies the mathematical foundations of cryptographic methods.

11.6.2 Basic Cryptographic Algorithms

A method of encryption and decryption is called a cipher. Some cryptographic methods rely on the secrecy of the algorithms; such algorithms are only of historical interest and are not adequate for real-world needs. All modern algorithms use a key to control encryption and decryption; a message can be decrypted only if the key matches the encryption key. The key used for decryption can be different from the encryption key, but for most algorithms they are the same.

There are two classes of key-based algorithms, symmetric (or secret-key) and asymmetric (or public-key) algorithms. The difference is that symmetric algorithms use the same key for encryption and decryption (or the decryption key is easily derived from the encryption key), whereas asymmetric algorithms use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key.

Symmetric algorithms can be divided into stream ciphers and block ciphers. Stream ciphers can encrypt a single bit of plaintext at a time, whereas block ciphers take a number of bits (typically 64 bits in modern ciphers), and encrypt them as a single unit.

Asymmetric ciphers (also called public-key algorithms or generally public-key cryptography) permit the encryption key to be public (it can even be published in a newspaper), allowing anyone to encrypt with the key, whereas only the proper recipient (who knows the decryption key) can decrypt the message. The encryption key is also called the public key and the decryption key the private key or secret key.

Modern cryptographic algorithms cannot really be executed by humans. Strong cryptographic algorithms are designed to be executed by computers or specialized hardware devices. In most applications, cryptography is done in computer software, and numerous cryptographic software packages are available.

Generally, symmetric algorithms are much faster to execute on a computer than asymmetric ones. In practice they are often used together, so that a public-key algorithm is used to encrypt a randomly generated encryption key, and the random key is used to encrypt the actual message using a symmetric algorithm.

Many good cryptographic algorithms are widely and publicly available in any major bookstore, scientific library, or patent office, and on the Internet. Well-known symmetric functions include DES and IDEA. RSA is probably the best known asymmetric algorithm.

11.6.3 Digital Signatures

Some public-key algorithms can be used to generate digital signatures. A digital signature is a block of data that was created using some secret key, and there is a public key that can be used to verify that the signature was really generated using the corresponding private key. The algorithm used to generate the signature must be such that without knowing the secret key it is not possible to create a signature that would verify as valid.

Digital signatures are used to verify that a message really comes from the claimed sender (assuming only the sender knows the secret key corresponding to his/her public key). They can also be used to timestamp documents: a trusted party signs the document and its timestamp with his/her secret key, thus testifying that the document existed at the stated time.

Digital signatures can also be used to testify (or certify) that a public key belongs to a particular person. This is done by signing the combination of the key and the information about its owner by a trusted key. The reason for trusting that key may again be that it was signed by another trusted key. Eventually some key must be a root of the trust hierarchy (that is, it is not trusted because it was signed by somebody, but

because you believe a priori that the key can be trusted). In a centralized key infrastructure there are very few roots in the trust network (e.g., trusted government agencies; such roots are also called certification authorities). In a distributed infrastructure there need not be any universally accepted roots, and each party may have different trusted roots (such of the party's own key and any keys signed by it). This is the web of trust concept used e.g. in PGP.

A digital signature of an arbitrary document is typically created by computing a message digest from the document, and concatenating it with information about the signer, a timestamp, etc. The resulting string is then encrypted using the private key of the signer using a suitable algorithm. The resulting encrypted block of bits is the signature. It is often distributed together with information about the public key that was used to sign it. To verify a signature, the recipient first determines whether it trusts that the key belongs to the person it is supposed to belong to (using the web of trust or a priori knowledge), and then decrypts the signature using the public key of the person. If the signature decrypts properly and the information matches that of the message (proper message digest etc.), the signature is accepted as valid.

Several methods for making and verifying digital signatures are freely available. The most widely known algorithm is RSA.

11.6.4 Cryptographic Hash Functions

Cryptographic hash functions are typically used to compute the message digest when making a digital signature. A hash function compresses the bits of a message to a fixed-size hash value in a way that distributes the possible messages evenly among the possible hash values. A cryptographic hash function does this in a way that makes it extremely difficult to come up with a message that would hash to a particular hash value.

Cryptographic hash functions typically produce hash values of 128 or more bits. This number is vastly larger than the number of different messages likely to ever be exchanged in the world.

Many good cryptographic hash functions are freely available. Well-known ones include MD5 and SHA.

11.6.5 Cryptographic Random Number Generators

Cryptographic random number generators generate random numbers for use in cryptographic applications, such as for keys. Conventional random number generators available in most programming languages or programming environments are not suitable for use in cryptographic applications (they are designed for statistical randomness, not to resist prediction by cryptanalysts).

In the optimal case, random numbers are based on true physical sources of randomness that cannot be predicted. Such sources may include the noise from a semiconductor device, the least significant bits of an audio input, or the intervals between device interrupts or user keystrokes. The noise obtained from a physical source is then "distilled" by a cryptographic hash function to make every bit depend on every other bit. Quite often a large pool (several thousand bits) is used to contain randomness, and every bit of the pool is made to depend on every bit of input noise and every other bit of the pool in a cryptographically strong way.

When true physical randomness is not available, pseudo-random numbers must be used. This situation is undesirable, but often arises on general purpose computers. It is always desirable to obtain some environmental noise - even from device latencies, resource utilization statistics, network statistics, keyboard interrupts, or whatever. The point is that the data must be unpredictable for any external observer; to achieve this, the random pool must contain at least 128 bits of true entropy.

Cryptographic pseudo-random generators typically have a large pool ("seed value") containing randomness. Bits are returned from this pool by taking data from the pool, optionally running the data through a cryptographic hash function to avoid revealing the contents of the pool. When more bits are needed, the pool is stirred by encrypting its contents by a suitable cipher with a random key (that may be taken from an unreturned part of the pool) in a mode which makes every bit of the pool depend on every other bit of the pool. New environmental noise should be mixed into the pool before stirring to make predicting previous or future values even more impossible.

Even though cryptographically strong random number generators are not very difficult to built if designed properly, they are often overlooked. The importance of the random number generator must thus be

emphasized - if done badly, it will easily become the weakest point of the system.

11.6.6 Strength of Cryptographic Algorithms

Good cryptographic systems should always be designed so that they are as difficult to break as possible. It is possible to build systems that cannot be broken in practice (though this cannot usually be proved). This does not significantly increase system implementation effort; however, some care and expertise is required. There is no excuse for a system designer to leave the system breakable. Any mechanisms that can be used to circumvent security must be made explicit, documented, and brought into the attention of the end users.

In theory, any cryptographic method with a key can be broken by trying all possible keys in sequence. If using brute force to try all keys is the only option, the required computing power increases exponentially with the length of the key. A 32 bit key takes 2^32 (about 10^9) steps. This is something any amateur can do on his/her home computer. A system with 40 bit keys (e.g. US-exportable version of RC4) takes 2^40 steps - this kind of computing power is available in most universities and even smallish companies. A system with 56 bit keys (such as DES) takes a substantial effort, but is quite easily breakable with special hardware. The cost of the special hardware is substantial but easily within reach of organized criminals, major companies, and governments. Keys with 64 bits are probably breakable now by major governments, and will be within reach of organized criminals, major companies, and lesser governments in a few years. Keys with 80 bits may become breakable in future. Keys with 128 bits will probably remain unbreakable by brute force for the foreseeable future. Even larger keys are possible; in the end we will encounter a limit where the energy consumed by the computation, using the minimum energy of a quantum mechanic operation for the energy of one step, will exceed the energy of the mass of the sun or even of the universe.

However, key length is not the only relevant issue. Many ciphers can be broken without trying all possible keys. In general, it is very difficult to design ciphers that could not be broken more effectively using other methods. Designing your own ciphers may be fun, but it is not recommended in real applications unless you are a true expert and know exactly what you are doing.

One should generally be very wary of unpublished or secret algorithms. Quite often the designer is then not sure of the security of the algorithm, or its security depends on the secrecy of the algorithm. Generally, no algorithm that depends on the secrecy of the algorithm is secure. Particularly in software, anyone can hire someone to disassemble and reverse-engineer the algorithm. Experience has shown that a vast majority of secret algorithms that have become public knowledge later have been pitifully weak in reality.

The key lengths used in public-key cryptography are usually much longer than those used in symmetric ciphers. There the problem is not that of guessing the right key, but deriving the matching secret key from the public key. In the case of RSA, this is equivalent to factoring a large integer that has two large prime factors. In the case of some other cryptosystems it is equivalent to computing the discrete logarithm modulo a large integer (which is believed to be roughly comparable to factoring). Other cryptosystems are based on yet other problems.

To give some idea of the complexity, for the RSA cryptosystem, a 256 bit modulus is easily factored by ordinary people. 384 bit keys can be broken by university research groups or companies. 512 bits is within reach of major governments. Keys with 768 bits are probably not secure in the long term. Keys with 1024 bits and more should be safe for now unless major algorithmic advances are made in factoring; keys of 2048 bits are considered by many to be secure for decades.

It should be emphasized that the strength of a cryptographic system is usually equal to its weakest point. No aspect of the system design should be overlooked, from the choice algorithms to the key distribution and usage policies.

11.6.7 Cryptanalysis and Attacks on Cryptosystems

Cryptanalysis is the art of deciphering encrypted communications without knowing the proper keys. There are many cryptanalytic techniques. Some of the more important ones for a system implementor are described below.

Ciphertext-only attack: This is the situation where the attacker does not know anything about the contents of the message, and must work from ciphertext only. In practice it is quite often possible to make guesses about the plaintext, as many types of messages have fixed format headers. Even ordinary letters

and documents begin in a very predictable way. It may also be possible to guess that some ciphertext block contains a common word.

Known-plaintext attack: The attacker knows or can guess the plaintext for some parts of the ciphertext. The task is to decrypt the rest of the ciphertext blocks using this information. This may be done by determining the key used to encrypt the data, or via some shortcut.

Chosen-plaintext attack: The attacker is able to have any text he likes encrypted with the unknown key. The task is to determine the key used for encryption. Some encryption methods, particularly RSA, are extremely vulnerable to chosen-plaintext attacks. When such algorithms are used, extreme care must be taken to design the entire system so that an attacker can never have chosen plaintext encrypted.

Man-in-the-middle attack: This attack is relevant for cryptographic communication and key exchange protocols. The idea is that when two parties are exchanging keys for secure communications (e.g., using Diffie-Hellman), an adversary puts himself between the parties on the communication line. The adversary then performs a separate key exchange with each party. The parties will end up using a different key, each of which is known to the adversary. The adversary will then decrypt any communications with the proper key, and encrypt them with the other key for sending to the other party. The parties will think that they are communicating securely, but in fact the adversary is hearing everything.

One way to prevent man-in-the-middle attacks is that both sides compute a cryptographic hash function of the key exchange (or at least the encryption keys), sign it using a digital signature algorithm, and send the signature to the other side. The recipient then verifies that the signature came from the desired other party, and that the hash in the signature matches that computed locally.

Timing Attack: This very recent attack is based on repeatedly measuring the exact execution times of modular exponentiation operations. It is relevant to at least RSA, Diffie-Hellman, and Elliptic Curve methods. More information is available in the original paper and various followup articles.

There are many other cryptographic attacks and cryptanalysis techniques. However, these are probably the most important ones for a practical system designer. Anyone contemplating to design a new encryption algorithm should have a much deeper understanding of these issues. One place to start looking for information is the excellent book Applied Cryptography by Bruce Schneier.

11.7 Cryptographic Algorithms

The section describes some of the better known cryptographic algorithms, and presents some details as to the operation of selected algorithms.

11.7.1 Public Key Algorithms

Public key algorithms use a different key for encryption and decryption, and the decryption key cannot (practically) be derived from the encryption key. Public key methods are important because they can be used to transmit encryption keys or other data securely even when the parties have no opportunity to agree on a secret key in private. All known methods are quite slow, and they are usually only used to encrypt session keys (randomly generated "normal" keys), that are then used to encrypt the bulk of the data using a symmetric cipher (see below).

11.7.1.1 RSA

RSA (Rivest-Shamir-Adelman) is the most commonly used public key algorithm. Can be used both for encryption and for signing. It is generally considered to be secure when sufficiently long keys are used (512 bits is insecure, 768 bits is moderately secure, and 1024 bits is good). The security of RSA relies on the difficulty of factoring large integers. Dramatic advances in factoring large integers would make RSA vulnerable. RSA is currently the most important public key algorithm. It is patented in the United States (expires year 2000), and free elsewhere.

At present, 512 bit keys are considered weak, 1024 bit keys are probably secure enough for most purposes, and 2048 bit keys are likely to remain secure for decades.

One should know that RSA is very vulnerable to chosen plaintext attacks. There is also a new timing attack that can be used to break many implementations of RSA. The RSA algorithm is believed to be safe when used properly, but one must be very careful when using it to avoid these attacks.

It works as follows: take two large primes, p and q, and find their product $n = pq$; n is called the modulus. Choose a number, e, less than n and relatively prime to $(p-1)(q-1)$, and find its inverse, d, $mod\,[(p-1)(q-1)]$, which means that $ed = 1\,mod\,[(p-1)(q-1)]$; e and d are called the public and private exponents, respectively. Two numbers are relatively prime if they have no prime factors in common. The public key is the pair (n,e); the private key is (n,d). The factors p and q must be kept secret, or destroyed.

Example:

$$\begin{aligned} p &= 37 \\ q &= 51 \\ Then\, n &= 1887 \\ (p-1)(q-1) &= 36 \times 50 \\ &= 1800 \\ &= 2 \times 2 \times 2 \times 3 \times 3 \times 5 \times 5 \\ Let\, e &= 7 \times 7 \times 13 \\ &= 637 \\ Find\, d &= 373\,(637 \times 373 = 237601 = 1800 \times 132 + 1) \end{aligned}$$

Thus the public key is $(1887, 637)$, and the private key is $(1887, 373)$. An small program for finding d is shown below:

```
long d = 0l;
while (d++)
  if (((d * 637) % 1800) == 1)
    printf ("%ld\n", d);
```

It is difficult (presumably) to obtain the private key (n,d) from the public key (n,e). If one could factor n into p and q, however, then one could obtain the private exponent d. Thus the entire security of RSA is predicated on the assumption that factoring is difficult; an easy factoring method would "break" RSA

Here is how RSA can be used for privacy and authentication (in practice, actual use is slightly different).

RSA privacy (encryption): suppose Alice wants to send a private message, m, to Bob. Alice creates the ciphertext c by exponentiating: $c = m^e\, mod\, n$, where e and n are Bob's public key. To decrypt, Bob also exponentiates: $m = c^d\, mod\, n$, and recovers the original message m; the relationship between e and d ensures that Bob correctly recovers m. Since only Bob knows d, only Bob can decrypt.

Example:

$$\begin{aligned} Alice\, sends\, m &= 42 \\ c &= 42^{637}\, mod\, 1887 \\ &= 315 \\ Bob\, decypts\, m &= 315^{373}\, mod\, 1887 \\ &= 42 \end{aligned}$$

RSA authentication: suppose Alice wants to send a signed document m to Bob. Alice creates a digital signature s by exponentiating: $s = m^d\, mod\, n$, where d and n belong to Alice's key pair. She sends s and m to Bob. To verify the signature, Bob exponentiates and checks that the message m is recovered: $m = s^e\, mod\, n$, where e and n belong to Alice's public key.

Thus encryption and authentication take place without any sharing of private keys: each person uses only other people's public keys and his or her own private key. Anyone can send an encrypted message or verify a signed message, using only public keys, but only someone in possession of the correct private key can decrypt or sign a message.

11.7.1.2 Diffie-Hellman

Diffie-Hellman is a commonly used public-key algorithm for key exchange. It is generally considered to be secure when sufficiently long keys and proper generators are used. The security of Diffie-Hellman relies on the difficulty of the discrete logarithm problem (which is believed to be computationally equivalent to factoring large integers). Diffie-Hellman is claimed to be patented in the United States, but the patent expires April 29, 1997. There are also strong rumors that the patent might in fact be invalid (there is evidence of it having been published over an year before the patent application was wiled).

Diffie-Hellman is sensitive to the choice of the strong prime and the generator. One possible prime/generator pair is suggested in the Photuris draft. The size of the secret exponent is also important for its security. Conservative advice is to make the random exponent twice as long as the intended session key.

One should note the results presented in Brian A. LaMacchia and Andrew M. Odlyzko, Computation of Discrete Logarithms in Prime Fields, Designs, Codes and Cryptography 1 (1991), 47-62. Basically, they conclude that by doing pre-computations, it is possible to compute discrete logarithms relative to a particular prime efficiently. The work needed for the pre-computation is approximately equal or slightly higher than the work needed for factoring a composite number of the same size. In practice this means that if the same prime is used for a large number of exchanges, it should be larger than 512 bits in size, preferably 1024 bits.

There is also a new timing attack that can be used to break many implementations of Diffie-Hellman.

11.7.1.3 LUC

LUC is a public key encryption system. It uses Lucas functions instead of exponentiation. It's inventor Peter Smith has since then implemented four other algorithms with Lucas functions: LUCDIF, a key negotiation method like Diffie-Hellman; LUCELG PK, equivalent to El Gamal public-key encryption; LUCELG DS, equivalent to El Gamal digital signature; and LUCDSA, equivalent to the US Digital Signature Standard. LUC Encryption Technology Ltd (LUCENT) has obtained patents for cryptographic use of Lucas functions in United States and New Zealand.

11.7.2 Secret Key Algorithms (Symmetric Ciphers)

Secret key algorithms use the same key for both encryption and decryption (or the other is easily derivable from the other).

11.7.2.1 DES

DES is an algorithm developed in the 1970s. It was made a standard by the US government, and has also been adopted by several other governments worldwide. It is widely used, especially in the financial industry.

DES is a block cipher with 64-bit block size. It uses 56-bit keys. This makes it fairly easy to break with modern computers or special-purpose hardware. DES is still strong enough to keep most random hackers and individuals out, but it is easily breakable with special hardware by government, criminal organizations, or major corporations. In large volumes, the cost of breaking DES keys is on the order of tens of dollars. DES is getting too weak, and should not be used in new designs.

A variant of DES, Triple-DES or 3DES is based on using DES three times (normally in an encrypt-decrypt-encrypt sequence with three different, unrelated keys). Many people consider Triple-DES to be much safer than plain DES.

DES processes plaintext blocks of $n = 64$ bits, producing 64 bit ciphertext blocks. The size of the secret key K is 56 bits, specified as 64 bits, 8 of which are used as parity bits. There is a belief that the parity bits were introduced to weaken DES, reducing the exhaustive key search by 256.

Encryption proceeds in 16 stages (rounds). For each round, a 48 bit sub-key K_i is generated from the input key K. Within each round, 8 fixed 6-to-4 bit substitution mappings (S_i - S boxes - collectively S) are used. The 64 bit plaintext is divided into 32 bit halves, L_0 and R_0. Each round takes the 32 bit inputs from the previous round and produces 32 bit outputs as follows:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i), \end{aligned}$$

where

$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$

Here E is a fixed expansion permutation mapping R_{i-1} from 32 to 48 bits. P is a fixed permutation on 32 bits. The operator $\oplus$represents exclusive or. An initial bit permutation precedes the first round; following the last round the left and right halves are exchanged and the resulting string bit permuted by the inverse of the initial permutation.

Decryption involves the same key and algorithm, but with sub-keys applied to the internal rounds in the reverse order.

11.7.2.2 IDEA

IDEA (International Data Encryption Algorithm) is an algorithm developed at ETH Zurich in Switzerland. It uses a 128 bit key, and it is generally considered to be very secure. It is currently one of the best public known algorithms. It is a fairly new algorithm, but it has already been around for several years, and no practical attacks on it have been published despite of numerous attempts to analyze it.

IDEA is patented in the United States and in most of the European countries. The patent is held by Ascom-Tech. Non-commercial use of IDEA is free. Commercial licenses can be obtained by contacting idea@ascom.ch.

11.7.2.3 RC4

RC4 is a cipher designed by RSA Data Security, Inc. It used to be a trade secret, until someone posted source code for an algorithm in Usenet News, claiming it to be equivalent to RC4. There is very strong evidence that the posted algorithm is indeed equivalent to RC4. The algorithm is very fast. Its security is unknown, but breaking it does not seem trivial either. Because of its speed, it may have uses in certain applications. It can also accept keys of arbitrary length. RC4 is essentially a pseudo random number generator, and the output of the generator is XORed with the data stream. For this reason, it is very important that the same RC4 key never be used to encrypt two different data streams.

Source code and information about RC4 can be found here and in many cryptographic libraries, e.g. SSLeay, Crypto++, and Ssh source code.

The United States government routinely approves RC4 with 40 bit keys for export. Keys that are this small can be easily broken by governments, criminals, and amateurs.

It is interesting to know that the exportable version of SSL (Netscape's Secure Socket Layer), which uses RC4-40, was recently broken by at least two independent groups. Breaking it took about eight days; in many major universities (or companies) the corresponding amount of computing power is available to any computer science major. More information about the incident can be found on Damien Doligez's SSL cracking page.

11.7.2.4 Skipjack

Skipjack is the encryption algorithm contained in the Clipper chip; it was designed by the NSA. It uses an 80-bit key to encrypt 64-bit blocks of data; the same key is used for the decryption. Skipjack can be used in the same modes as DES (see Question 5.3), and may be more secure than DES, since it uses 80-bit keys and scrambles the data for 32 steps, or "rounds"; by contrast, DES uses 56-bit keys and scrambles the data for only 16 rounds.

The details of Skipjack are classified. The decision not to make the details of the algorithm publicly available has been widely criticized. Many people are suspicious that Skipjack is not secure, either due to oversight by its designers, or by the deliberate introduction of a secret trapdoor. By contrast, there have been many attempts to find weaknesses in DES over the years, since its details are public. These numerous attempts (and the fact that they have failed) have made people confident in the security of DES. Since

Skipjack is not public, the same scrutiny cannot be applied toward it, and thus a corresponding level of confidence may not arise.

Clipper is an encryption chip developed and sponsored by the U.S. government as part of the Capstone project. Announced by the White House in April, 1993, Clipper was designed to balance the competing concerns of federal law-enforcement agencies with those of private citizens and industry. The law-enforcement agencies wish to have access to the communications of suspected criminals, for example by wire-tapping; these needs are threatened by secure cryptography. Industry and individual citizens, however, want secure communications, and look to cryptography to provide it.

Clipper technology attempts to balance these needs by using escrowed keys. The idea is that communications would be encrypted with a secure algorithm, but the keys would be kept by one or more third parties (the "escrow agencies"), and made available to law-enforcement agencies when authorized by a court-issued warrant. Thus, for example, personal communications would be impervious to recreational eavesdroppers, and commercial communications would be impervious to industrial espionage, and yet the FBI could listen in on suspected terrorists or gangsters.

Each chip also contains a unique 80-bit unit key U, which is escrowed in two parts at two escrow agencies; both parts must be known in order to recover the key. Also present is a serial number and an 80-bit "family key" F; the latter is common to all Clipper chips. The chip is manufactured so that it cannot be reverse engineered; this means that the Skipjack algorithm and the keys cannot be read off the chip.

When two devices wish to communicate, they first agree on an 80-bit '"session key"' K. The method by which they choose this key is left up to the implementor's discretion; a public-key method such as RSA or Diffie-Hellman seems a likely choice. The message is encrypted with the key K and sent; note that the key K is not escrowed. In addition to the encrypted message, another piece of data, called the law-enforcement access field (LEAF), is created and sent. It includes the session key K encrypted with the unit key U, then concatenated with the serial number of the sender and an authentication string, and then, finally, all encrypted with the family key. The exact details of the law-enforcement field are classified.

The receiver decrypts the law-enforcement field, checks the authentication string, and decrypts the message with the key K.

11.7.2.5 Enigma

Enigma was the cipher used by the Germans in World War II. It is trivial to solve with modern computers; see the Crypt Breaker's Workbench tool. This cipher is used by the Unix crypt(1) program, which should thus not be used.

11.7.3 Block Cipher Modes

Many commonly used ciphers (e.g., IDEA, DES, BLOWFISH) are block ciphers. This means that they take a fixed-size block of data (usually 64 bits), an transform it to another 64 bit block using a function selected by the key. The cipher basically defines a one-to-one mapping from 64-bit integers to another permutation of 64-bit integers.

If the same block is encrypted twice with the same key, the resulting ciphertext blocks are the same (this method of encryption is called Electronic Code Book mode, or ECB). This information could be useful for an attacker.

In practical applications, it is desirable to make identical plaintext blocks encrypt to different ciphertext blocks. Two methods are commonly used for this:

- CFB mode: a ciphertext block is obtained by encrypting the previous ciphertext block, and XORing the resulting value with the plaintext.
- CBC mode: a ciphertext block is obtained by first XORing the plaintext block with the previous ciphertext block, and encrypting the resulting value.

The previous ciphertext block is usually stored in an Initialization Vector (IV). An initialization vector of zero is commonly used for the first block, though other arrangements are also in use.

11.7.4 Privacy in the Internet

Every time we use the Internet, either for surfing the World Wide Web or sending E-Mails, we leave our traces behind. These traces can be analyzed, and a lot of information can be taken from them.

11.7.4.1 How your personal information Gets collected

Whenever you connect to a web server, view a web site, or send an E-Mail, The servers along the way log these activities. The information can be collected in two ways :

- Direct disclosure of information : Many sites, make you register them before you can enter. Even if the registration is free, you are needed to give your information. Your name, E-mail, address, and whatever they site wants. The server remembers these pieces of information. You can never be sure as to what will be done with that information. But at least you're aware of that information collection, and if you wish to, you can simply choose not to give the info, and not to enter the site.

- Passive recording of information : When you visit a site, the server logs your entry. It can know all sorts of things about you, just from looking at your messages, even if they are encrypted! the message headers cannot be encrypted, as they need to be routed through the internet. Your mailing address, and IP can always be extracted. Your own server can also record stuff you might not with him too (Your boss might look at that log, or your spouse might...). It knows where you surf, what you look at, and whom you E-mail. The Web browser you use can also know a great deal about you : And that information can sometimes be extracted from the browser...

11.7.4.2 What information can be revealed about you

A web site with the right equipment, can know a great deal. The information that can be revealed includes your E-mail address, your IP address, the files you viewed, and the pages you visited.

When you send E-mail, you're actually getting quite exposed. You can encrypt the message body, but you can't hide the headers, if you want the message to travel through the net. Let's look at an example E-Mail header, taken from a Netscape Mailer :

```
Return-Path: <yogevm@math.tau.ac.il>
Received: from bfmail4 ([206.156.198.174]) by e4000.artaxia.com (8.8.5/8.8.5) with
  SMTP id TAA08477 for <merterc@artaxia.com>; Thu, 5 Jun 1997 19:28:01 -0200 (GMT)
Received: from taurus.math.tau.ac.il (132.67.64.4) by bfmail4.bigfoot.com with SMTP
  ( Bigfoot SMTP Server May 8 1997 15:22:04 ); Thu, 05 Jun 1997 12:25:22 -400
  (Eastern Standard Time)
Received: from lune.math.tau.ac.il (yogevm@lune.math.tau.ac.il [132.67.96.11])
  by taurus.math.tau.ac.il (8.8.3/8.8.3) with SMTP id TAA23843;
  Thu, 5 Jun 1997 19:22:21 +0300 (GMT+0300)
Date: Thu, 5 Jun 1997 19:22:20 +0300 (GMT+0300)
From: Mashiach Yogev <yogevm@math.tau.ac.il>
To: Mertens Ron <merterc@bigfoot.com>
Subject: Re: [Fwd: Re: "Operating Systems" - The Exam]
In-Reply-To: <3396C160.7745@netvision.net.il> Message-ID:
  <Pine.SUN.3.95.970605192120.27235C-100000@lune.mat h.tau.ac .il>
  MIME-Version: 1.0 Content-Type: TEXT/PLAIN; charset=US-ASCII X-UIDL:
  011a6057e9a080092b8d36ce7f0fc9e8
Status: U X-PMFLAGS: 36176000 0
```

We can see who sent this message (Mashiach Yogev) and we could also find his Email address. The message was sent to Ron Mertens. We can see the subject (an Exam in Operation Systems), the Date it was sent, and even the path the message went through to get to it's destination.

When you connect to a web server, things are much worse than that. Let's take a look at the NCSA server (it's quite popular). It includes a program called httpd. It maintains 3 log files :

- access_log : it logs every access to the server. The name of the accessing site, the requested file name, time of access and some more info

- error_log : Files that were requested, that doesn't exist
- refer_log : The links that point to the links on the server agent_log : A list of the programs that have contacted the server.

The 'problem' lies within the HTTP protocol. It has some features that allow all that data to be collected. The TCP/IP protocol, has a sort of caller-ID build in. When you connect, you send your computer's name, and the IP address. The refer_log is also a problem. It is used mainly for advertisement : for companies to be able to focus the advertisement more correctly, but it can be used for other means!

A newer source of trouble, are the 'Cookies'. Cookies are client side persistent information. Almost all of the new browsers have this facility : It allows web sites to store information about your visit, in your own hard disk. When you enter the site again, it will read your cookie, and thus now that you've been there already. It is used for nice tricks such as a personalized web page, and so on, but it can a serious privacy breach.

If you're connected to the internet through a Proxy, you have still another problem. The proxy server logs every access to the outside web, by every member of the organization. Your IP address, and your host computer are written down as well.

11.7.4.3 Why should I care about all that?

Well, you Should. Your privacy is your Right, as a human. If your privacy is violated, other freedoms (like the freedom of expression, or religion) might get threatened. Even if you have nothing to hide (like most people think), your privacy right must be important to you! Of course, it seems safe to surf the internet, but be warned that your are watched. Your privacy is not guaranteed, and that's a problem. Most of the information that can be collected is never used, but it can be. And there's not much you can do about it. A detailed profile of yourself can be created, and your tastes and preferences can be learned. Just by recording your message, and links, and web activity. This information can fall to the wrong hands (marketers and governments, for example) and be used against your interests.

11.7.4.4 How can I assure my privacy?

You can wait for your country to legislate a law about privacy, although these things take time, and most likely will never happen. In the meantime, you can use several offered utilities to ensure your privacy :

- Anonymous Remailers : remailers are program that route E-mail (or Usenet message, for that matter) and posts them anonymously. The recipient cannot determine who sent the E-mail. There are different classes of these program, but most of them do the same things. Some of these programs also combine the privacy with encryption (using PGP) and so they are very useful, if you find your privacy or security shattered.
- Cookies Clearers : Because the cookies are saved on your local disk, it's actually quite easy to delete them if you feel they are a threat to your privacy! there are programs that do that in a safe and clean way.
- The Anonymizer : This is a web site, created by Community ConneXion for those who are really worried. It is a web site, that shields your information from other sites. When you visit the Anonymizer, you are given an anonymous identity, and when you access other sites, they receive that identity and not your true one. It works even if you follow links to other sites, and it is a good privacy provider!
- Anonymous Shell Accounts : Your Internet Provider knows a great deal about you. Every action you perform while you're on-line is known to your ISP. Several providers allow you to open an anonymous account, and thus be protected (from themselves...).
- Electronic-Cash : In order to secure your privacy when buying on-line, the idea of electronic cash came to life. The basic idea is having your money in your computer : You'll withdraw money from

your account, and then you can spend it on line. With the E-cash system, when you buy, your identity is not revealed automatically. If you wish to protect your privacy, you can remain anonymous. It's good for on-line services, which don't really require your name.

11.7.4.5 Problems with privacy control

The main problem is that the internet is a world-wide network. People from all around the world visit it, and so it's hard to enforce laws on it. In the US, for example there is no comprehensive law that protects people's privacy. There are several guidelines that protect some areas of your privacy, but it's not enough. And most countries are falling behind the US in that area. Another problem is that the web browsers, are usually made by the same companies that make the web servers. The ability to collect information about the site visitors is a major selling points, and your privacy can be endangered by this fact.

Appendix A

Practical Exercises

A.1 Outcomes

1. Demonstrate practical knowledge and understanding of the information, concepts and principles applicable to computer networking.
2. Collect, present and analyze data relating to the performance of computer networks.
3. Work effectively with others situated on remote workstations, using a computer network.
4. Communicate an understanding of computer networking and the results of laboratory work.
5. Use computer networking technology with due appreciation of the social (particularly security) issues involved.
6. Ability to communicate ideas relating to computer networks, specifically protocol designs, network layout and network models in appropriate written and diagrammatic form.
7. Demonstrate the ability to access new information from appropriate sources, particularly the Internet, and reference such material correctly.
8. Incorporate such new information into solutions to problems involving computer networks.
9. Appreciate that there is a need for life-long learning to remain current in the field of computer networking.

A.2 Objectives

These exercises:

1. Show how real network traffic can be monitored, illustrating layering of protocols and the protocol headers with realistic values.
2. Demonstrate the construction of a protocol which resolves a particular communication issue.
3. Require the collection and analysis of network traffic patterns.
4. Model a number of network scenarios and permit abstract evaluation of their characteristics.
5. Investigate security and social implications of the development of network applications.
6. Require the use of concepts and knowledge presented during the course.

7. Involve the use of more than one workstation, using networked communication, for effective exploration of the communication issues.
8. Require that the results be clearly presented in a neat and understandable form, without ambiguity.
9. Can benefit from additional reading, provided that it is suitably referenced.
10. Reward innovation that still solves the original problem but in an innovative way.

A.3 Practical Exercises

Please see the course web page for details of the practical exercises.

Appendix B

Glossary of Terms and Acronyms

B.1 Acronyms

A:-

AAL ATM Adaptation Layer

ADSL Asymmetrical Digital Subscriber Line

ANSI American National Standards Institute

ARCNET Attached Resource Computer Network. A 2.5-100Mbps token bus LAN developed by Datapoint Corp. It is simple, easy to use and inexpensive.

ARP Address Resolution Protocol (Layer 3)

ASN.1 Abstract Syntax Notation 1 (ISO 8824 8825)

ATM Asynchronous Transmission Mode

AUI Attachment Unit Interface cable (15 pin DIX)

B:-

B-ISDN Broad band Integrated Services Digital Network

BNC British Naval Connector or Barrel Network Connector (10Base2)

BRI Basic Rate Interface (ISDN: 2x64+16 Kbps)

C:-

CBR Constant Bit Rate

CMIP Common Management Information Protocol (ISO)

CRC Cyclic Redundancy Check

CSMA/CD Carrier Sense Multiple Access with Collision Detection is the IEEE 802.3 MAC Layer protocol.

D:-

DNS Domain Name System (Internet)

E:-

F:-

FAQ Frequently Asked (Answered) Questions

FCS Frame Check Sequence
FDDI Fiber Distributed Data Interchange
FTP File Transfer Protocol

G:-

H:-

I:-

ICMP Internet Control Message Protocol
IEEE Institute of Electrical and Electronics Engineers
IMAP Internet Message Access Protocol
IP Internet Protocol (layer 3)
IPng Internet Protocol Next Generation (IETF)
IPv6 Internet Protocol Version 6
IPX Internet Packet Exchange (Netware, layer 3), Novell's Internetwork Packet Exchange protocol
ISDN Integrated Services Digital Network

J:-

K:-

L:-

LAN Local Area Network
LLC Logical Link Control (IEEE 802.2)

M:-

MAC Media Access Control protocol (IEEE 802.3, 802.5, etc.)
MAN Metropolitan Area Network

N:-

NETBEUI Network Basic Easy User Interface - Microsoft's version of the IEEE 802.3 protocol.
NETBIOS Network Basic Input/Output System.
NIC Network Interface Card

O:-

OC-1 Optical Carrier level 1 (51.84 Mb/s)
OC-3 Optical Carrier level 3 (155 Mb/s)
OC-12 Optical Carrier level 12 (622 Mb/s)
OC-48 Optical Carrier level 12 (2.4 Gb/s)
ODI Open Data Link Interface (Netware)
OSI Open Systems Interconnect (ISO 7498)

P:-

POP Post Office Protocol (RFC 1125)
PPP Point to Point Protocol

PRI Primary Rate Interface (ISDN)

Q:-

R:-

RIP Router Information Protocol

RJ-45 An 8 pin telephone type connector

RMON Remote MONitoring

RSA Rivest Shamir & Adleman (encryption)

RSVP resource ReSerVation Protocol

RTP Real-Time Transport Protocol

S:-

SAP Services Advertising Protocol (Netware)

SDH Synchronous Digital Hierarchy

SLIP Serial Line Internet Protocol

SNMP Simple Network Management Protocol

SMTP Simple Message Transfer Protocol (Internet - RFC 822)

STP Shielded Twisted Pair

T:-

T1 Trunk 1 (1544 bps, 24-channel PCM)

T3 Trunk 3 (44.736 Mb/s)

TCP Transmission Control Protocol

TELNET Standard protocol for logging into hosts using TCP/IP protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TFTP Trivial File Transfer Protocol

U:-

UDP User Datagram Protocol (layer 4)

UTP Unshielded Twisted Pair

V:-

W:-

WAN Wide Area Network

X:-

Y:-

Z:-

B.2 Networking Terms

0-9:-

10XYZ: These are the IEEE names for the different physical types of Ethernet. The "10" stands for signalling speed: 10MHz. "Base" means Baseband, "broad" means broadband. Initially, the last section as intended to indicate the maximum length of an unrepeated cable segment in hundreds of meters. This convention was modified with the introduction of 10BaseT, where the T means twisted pair, and 10BaseF where the F means fiber (see the following Q&A for specifics). This actually comes from the IEEE committee number for that media.
In actual practice:

- 10Base2 Is 10MHz Ethernet running over thin, 50 Ohm baseband coaxial cable.
- 10Base2 is also commonly referred to as thin-Ethernet or Cheapernet.
- 10Base5 Is 10MHz Ethernet running over standard (thick) 50 Ohm baseband coaxial cabling.
- 10BaseF Is 10MHz Ethernet running over fiber-optic cabling.
- 10BaseT Is 10MHz Ethernet running over unshielded, twisted- pair cabling.
- 10Broad36 Is 10MHz Ethernet running through a broadband cable.

10BASE2 IEEE 802.3 10BASE2: IEEE's standardized version of Digital Equipment Corporation's ThinWire Ethernet, which runs over a thinner coaxial cable than the original 10BASE5. Another old nickname for it was "CheaperNet". The coaxial cable is specific to 10BASE2, but two variants of off-the-shelf RG cable are sometimes used. The cable looks very similar to the cable used for IBM 3270-style terminals or that used for home cable TV, but has different electrical characteristics. The "2" in the name refers to the 200 meter (or more precisely, 185 meter) limit on the cable length. Like 10BASE5, computers are attached along the length of the cable.

10BASE-F Three variants of IEEE 802.3 which runs over multimode fiber.

10BASE-T A variant of IEEE 802.3 which allows stations to be attached via twisted-pair cable.

100VG-AnyLAN A 100 Mbps DPAM for integrated Ethernet and Token Ring transmission.

A:-

ATM: Asynchronous Transfer Mode A high speed connection oriented switching and multiplexing technology that uses 53 byte cells (5 byte header, 48 byte payload) to transmit different types of traffic simultaneously, including voice, video and data. It is asynchronous in that information streams can be sent independently without a common clock.

B:-

Bandwidth: A term used to indicate transmission capacity in “Hertz”. It represents the difference between the highest and lowest frequencies available for signal transmission. In a LAN, we usually speak of bandwidth in megabits per second, e.g. 10MBPS, 4 or 16MBPS (Token Ring), 100MBPS (Fast Ethernet), etc.

Baseband: A baseband network is one that provides a single channel for communications across the physical medium (e.g., cable), so only one device can transmit at a time. Devices on a baseband network, such as Ethernet, are permitted to use all the available bandwidth for transmission, and the signals they transmit do not need to be multiplexed onto a carrier frequency. An analogy is a single phone line such as you usually have to your house: Only one person can talk at a time–if more than one person wants to talk everyone has to take turns.

Bastion host: A computer system that must be highly secured because it is vulnerable to attack, usually because it is exposed to the Internet and is a main point of contact for users of internal networks. It gets its name from the highly fortified projections on the outer walls of medieval castles.

Bridge: an active device that takes a packet of data in one port, examines the packet to be sure that it is valid, and passes it to its other port(s) only if the destination device is not on the network segment attached to the port it came in. A bridge looks at the MAC address of the destination machine to make the decision of whether or not to forward the packet, and does not look at any of the data or protocol information in the packet. There is no rule on how many bridges can be on a network, and a bridge breaks up a network, so that there can be 4 repeaters on each port of a bridge.

Bridge: A network "relay" which reads, buffers, and sends data to relay it from one data link to another, but makes the two data links appear as one to levels higher than the data link layer.

Broadband: It is the opposite of a baseband network. With broadband, the physical cabling is virtually divided into several different channels, each with its own unique carrier frequency, using a technique called "frequency division modulation". These different frequencies are multiplexed onto the network cabling in such a way to allow multiple simultaneous "conversations" to take place. The effect is similar to having several virtual networks traversing a single piece of wire. Network devices "tuned" to one frequency can't hear the "signal" on other frequencies, and visa-versa. Cable-TV is an example of a broadband network: multiple conversations (channels) are transmitted simultaneously over a single cable; you pick which one you want to listen to by selecting one of the frequencies being broadcast.

Broadcast: A message (e.g. packet or frame) sent to all the nodes on a network.

Broadcast Address: An address which can be used as the destination of a communication that indicates the packet/message is a broadcast. IP has broadcast addresses as does IEEE 802.

Bus: For the purposes of LANs, bus is a term for a LAN topology which has the same characteristic: the same wire is attached to a number of devices which all share that wire to transmit to other devices on the LAN. ThickWire and ThinWire Ethernet, Localtalk, and ARCnet are examples of LAN technologies with a bus topology.

C:-

Client: any machine that request something from a server. The server supplies files and sometimes processing power to the smaller machines connected to it. Each machine is a client in this type of network.

Concentrator: a device which allows a number of stations to be connected to a LAN. In the case of Ethernet, it is simply a multi-port repeater. In the case of ring networks like Token Ring and FDDI, it acts as a switch which keeps the ring intact even if individual devices are unplugged.

CSMA/CD: "Carrier Sense Multiple Access with Collision Detection" The method by which nodes on an Ethernet/IEEE 802.3 LAN gain access to the network, i.e. one of several techniques that have been built into different LAN technologies to allow multiple nodes to share the same wires/electronics to send their data.

D:-

Dual-homed host: A general-purpose computer system that has at least two network interfaces (or homes)

E:-

Ethernet: LAN Data Link protocol developed by a consortium of vendors; later standardized as IEEE 802.3 with a few modifications. For many applications, users have not adopted all the IEEE 802.3 differences. Ethernet/802.3 now can be run on two types of coaxial cable as well as multi-mode fiber and unshielded twisted-pair. "Raw" rate of data transmission is 10 megabits/second.

F:-

Firewall: A component or set of components that restricts access between a protected network and the Internet, or between other sets of networks.

Full-duplex: Nodes may transmit and receive simultaneously.

G:-

Gateway: A type of "network relay" that attaches two networks to build a larger network. Modern "narrow" usage is that it is one that translates an entire stack of protocols, e.g., translates TCP/IP-style mail to ISO-style mail. Older usage used it for other types of relays–in particular, in the "TCP/IP" world, it has been used to refer to what many now insist is a "router".

Gateway (Software): software running on a computer that translates similar but different protocols. The computer may or may not have specialized hardware. An example would be a computer that translates Microsoft Mail to SMTP (Internet style) email. This may just be a PC-Clone 486 running a mail gateway and doing something else, or it could be a dedicated machine.

Gigabit Ethernet: High-speed version of Ethernet (a billion bits per second) under development by the IEEE.

H:-

Half-duplex: Only one node may send at a given time, and nodes take turns transmitting.

Header: A portion of a message (cell, packet, frame, etc) at the front with control information such as the destination address.

Hub: a nebulous term, typically applied to a multiport repeater or concentrator consisting of a chassis with slots to be populated by cards, allowing it to be configured with various numbers and combinations of LAN ports. Vendors of networking equipment often also have other types of devices that can be inserted in the slots such as terminal servers, bridges, routers, gateways, etc.

I:-

IEEE 802.2: An IEEE standard for the portion of LAN Data Link protocols that is the same for all flavors of IEEE LAN protocols, e.g. 802.3 and 802.5. Sometimes not used.

IEEE 802.3: An IEEE standard for LANs–their "improved" version of Ethernet. See Ethernet.

IEEE 802.4: An IEEE standard for LANs: Token Bus networks. Basically, standardizes MAP, a protocol that operates a Token Bus protocol on broadband.

IEEE 802.5: An IEEE standard for Token-Ring-based LANs. There are two types: 4Mbps and 16Mbps.

IP: "Internet Protocol" The basic protocol of TCP/IP and the Internet.

IPX: Novell's protocol used by Netware. Utilizes part of XNS. A router with "IPX routing" purports to interconnect LANs so that Novell Netware clients & servers can talk through the router.

ISDN: Integrated Services Digital Network - A switched digital transmission service provided by a local telco's switching office. Uses same copper as analog service so is practical for home, small office, school applications. Available in BRI (2 64kb data channels 1 signalling channel) or PRI (23 bearer(data/voice) channels 1 signalling channel)

J:-

Jumper cable: (also called patch cable), a cable used to connect the jack of the patch panel to the jack on the hub or repeater. Also used to connect from the jack in the room to the jack on the card. The most common kind of patch cable for 10 base-T is a "straight- through" cable. It is made of twisted pair wire with two eight pin modular connectors on it. For 10 base-T it must have at least two pair of cable, one pair hooking pin 1 at one end to pin 1 at the other end, and pin 2 to pin 2. The other pair connects pin 3 to pin 3 and pin 6 to pin 6. If the cable has two other pair that connects pins 4, 5, 7, and 8 that is OK but not required.

K:-

L:-

M:-

N:-

Network Interface Card (NIC): a card that goes in a device that allows it to connect directly to the network. Examples are Ethernet and token ring cards. Each NIC has an address built onto the card at the factory that makes it unique on a network. This address is often called a hardware address or a MAC (Media Access Control) Address.

Network Operating System (NOS): controls the interaction between all the machines on the network. The network operating system is responsible for controlling the way information is sent over the network medium and handles the way data from one machine is packaged and send to another. The NOS also has to handle what happens when two or more machines try to send at the same time.

O:-

OSI Reference Model A model put forth by the ISO for communication between computer equipment and networks, which maps out 7 protocol layers. This model explains what each layer does. The model is often used to explain anyones protocols (not just OSI) to the point where many people seem to believe that true data-communications requires these 7 layers.

Top Layer	7	Application
	6	Presentation
	5	Session
	4	Transport
	3	Network
	2	Data link (IEEE 802.x)
Bottom Layer	1	Physical (wire)

P:-

Packet filtering: The action a device takes to selectively control the flow of data to and from a network. Packet filters allow or block packets, usually while routing them from one network to another (most often from the Internet to an internal network, and vice versa). To accomplish packet filtering, you set up a set of rules that specify what types of packets (such as those to or from a particular IP address or port) are to be allowed and what types are to be blocked. Packet filtering may occur in a router, in a bridge, or on an individual host. It is sometimes known as screening.

Patch Panel: a passive device that allows direct connections to a room on one side, and a jack on the other. The jack is used to connect to a repeater or hub to a device attached to the jack in the room.

Perimeter network: A network added between a protected network and an external network, in order to provide an additional layer of security. A perimeter network is sometimes called a DMZ, which stands for De-Militarized Zone (named after the zone separating North and South Korea).

Protocol: The "rules" by which two network elements trade information in order to communicate. Must include rules about a lot of mundane detail as well as rules about how to recover from a lot of unusual communication problems. Thus they can be quite complicated.

Proxy server: A program that deals with external servers on behalf of internal clients. Proxy clients talk to proxy servers, which relay approved client requests on to real servers, and relay answers back to clients.

Q:-

R:-

Relay: One terminology uses the term "relay" as a device that interconnects LANs, different kinds of relays being repeaters, bridges, routers, and gateways.

Repeater: In the "Ethernet" world, a "relay" that regenerates and cleans up signals, but does no buffering of data packets. It can extend an Ethernet by strengthening signals, but timing limitations on Ethernets still limit their size.

Repeater: an active device that takes an electrical signal in one port, and sends the exact same data out its other port(s). It only looks at the electrical signals, not at the data contained in the signals. There is a limit in Ethernet that allows only 4 repeaters between any two end stations on a network.

RFC: "Request For Comments" The name is a real red herring when it comes to Internet RFCs. Some really are "Requests For Comments" but all Internet protocol documents are stamped with an RFC number that they never shake, so the acronym RFC generally refers to documents that describe protocols in the TCP/IP family.

Ring: A classification of network technology (known as its topology) exemplified by Token Ring and FDDI. The interconnected devices are connected one-to-another in the shape of a ring and data flows around it in one direction.

Router: software running in a specialized computer that has more than one NIC card. Routing software looks at protocol information in a packet of data to decide if that data should go to a network directly attached to that router, or a network further away. If it should go to one further away, the router figures out what other router is the next step in the path. Routers used on the global Internet route the Internet Protocol (IP) of the TCP/IP group of protocols. It is also possible to route other network layer protocols such as the DDP protocol in AppleTalk or the IPX protocol in Novell's Netware. Most company's selling routers sell both the hardware and the software all in one package.

Router: A network "relay" that uses a protocol beyond the Data Link protocol to route traffic between LANs and other network links.

Routing Protocol: a protocol sent between routers by which routers exchange information own how to route to various parts of the network. The TCP/IP family of protocols has a bunch, such as RIP, EGP, BGP, OSPF, and dual IS-IS.

S:-

Server: is any machine that can provide files, resources, or services to another machine. Any machine that you request a file from is a server. This is the essence of client-server networks: One machine, the client, request something from another machine, the server. A single machine may be both client and server. The more commonly used definition for a server is related to local area networks, where the server is a powerful machine that holds main files and large applications. Other machines on the network connect to the server to access those files and applications. In this type of network, a single machine usually acts as the server and all the other machines are clients. Simply put, the server is any machine on the network that your machine request something from.

Simplex: One node transmit exclusively, while another exclusively receives.

SNMP: "Simple Network Management Protocol" Originally developed to manage IP based network equipment like routers and bridges, now extended to wiring hubs, workstations, toasters, jukeboxes, etc. SNMP for IPX and AppleTalk under development. Widely implemented.

Star: A classification of network technology (known as its topology) defined by a network which consists of a central element attached to its client computers via wires leading out from the central element. A LAN that consists of a number of computers each directly attached to an ATM switch is a good example of a star-topology LAN.

Switch: an active device that has some of the features of a repeater and some of a bridge. Similar to a bridge, it gets the first part of a packet, and puts the data out only on the port which has the destination machine. Similar to a repeater, it does not wait for the entire packet, but starts sending the data out the correct port as soon as it can tell which is the right port. This is very much how a telephone switch places telephone calls. There is no rule about how many switches can be on a network, and similar to a bridge, a switch breaks up a network and allows 4 repeaters on each port.

Switched Ethernet: really the same as Ethernet as far as standards go: acts like a very fast multiport Ethernet bridge giving an Ethernet to each station. Presumably based on 10BASE-T for most stations.

T:-

TCP/IP: "Transmission Control Protocol/Internet Protocol" literally, two protocols developed for the Defense Data Network to allow their ARPANET to attach to other networks relatively transparently. The name also designates the entire family of protocols built out of IP and TCP. The Internet is based upon TCP/IP.

TELNET: a protocol in the TCP/IP family that is used for "remote login". The name is also often used as the name of the client program that utilizes the TELNET protocol.

ThickWire: "ThickWire" Ethernet or IEEE 802.3 10BASE5.

ThinWire: ThinWire Ethernet or IEEE 802.3 10BASE2.

Token Ring: People often use the term "Token Ring" to designate IEEE 802.5. In the more general sense of the phrase, a token ring is a type of LAN that has stations wired in a ring, where each station constantly passes a special message (a "token") on to the next. Whoever has the token can send a message.

Topology: Term used to describe a general characteristic of a LAN technology which more or less describes the shape of the necessary wiring. Three examples are bus, ring, and star.

Tunneling: An important concept in the design of many kinds of networks: taking some protocol-family's ability to move packets from user to user, or to open virtual-circuits between users, and use this as if it were a Data Link protocol to run another protocol family's upper layers (or even the same protocol family's upper layers). Examples: running TCP/IP over AppleTalk instead of something like Ethernet; running AppleTalk over DECNet instead of something like Localtalk or Ethernet.

Twisted Pair: wires that are grouped by pair such that each pair of wires is twisted together, causing an electrical effect that increases the ability of the wire to carry data over a distance.

U:-

V:-

W:-

X:-

Y:-

Z:-

Appendix C

Socket Programming

C.1 Socket Programming in Java

C.1.1 Networking Basics

Computers running on the Internet communicate to each other using the Internet Protocol (IP). Usually the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) can be found encapsulated within the Internet Protocol. Other higher level application protocols will generally be encapsulated inside these (such as HTTP which takes advantage of the reliable data-stream communication facilities provided by TCP).

When you write Java programs that communicate over the network, you are programming at the application layer. Typically, you don't need to concern yourself with the internals of the TCP and UDP layers. Instead, you can use the classes in the java.net package which allow your data to be encapsulated in either of these two protocols. These classes provide system-independent network communication. However, to decide which Java classes your programs should use, you do need to understand how TCP and UDP differ.

C.1.1.1 TCP

When two applications want to communicate to each other reliably, they establish a connection and send data back and forth over that connection. This is analogous to making a telephone call. If you want to speak to Aunt Beatrice in Kentucky, a connection is established when you dial her phone number and she answers. You send data back and forth over the connection by speaking to one another over the phone lines. Like the phone company, TCP guarantees that data sent from one end of the connection actually gets to the other end and in the same order it was sent. Otherwise, an error is reported.

Unlike a phone call, TCP still operates over the underlying unreliable packet based network. It uses connection establishment, positive acknowledgment with retransmission, and sequence numbers to create the appearance of a connection - a virtual circuit.

TCP provides a point-to-point channel for applications that require reliable communications. The Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Telnet are all examples of applications that require a reliable communication channel. The order in which the data is sent and received over the network is critical to the success of these applications. When HTTP is used to read from a URL, the data must be received in the order in which it was sent. Otherwise, you end up with a jumbled HTML file, a corrupt zip file, or some other invalid information.

Thus, choose TCP if:

- Reliability is an issue. You cannot afford to have any of the data lost or re-ordered.
- You are communicating between only two parties. Multiple TCP connections are required if more participants are involved, one for every pair of communicating stations.

- Time is not an issue. Providing reliable connections on an unreliable network requires extra signalling to detect lost or out-of-sequence packets, and retransmission to fill in missing pieces. This can slow communication.

C.1.1.2 UDP

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data, called datagrams, from one application to another. Sending datagrams is much like sending a letter through the postal service: The order of delivery is not important and is not guaranteed, and each message is independent of any other.

UDP provides minimal extra functionality over that provided at the IP layer. This is convenient when applications require little overhead and will not suffer from the lack of a reliable communication channel.

For many applications, the guarantee of reliability is critical to the success of the transfer of information from one end of the connection to the other. However, other forms of communication don't require such strict standards. In fact, they may be slowed down by the extra overhead or the reliable connection may invalidate the service altogether.

Consider, for example, a clock server that sends the current time to its client when requested to do so. If the client misses a packet, it doesn't really make sense to resend it because the time will be incorrect when the client receives it on the second try. If the client makes two requests and receives packets from the server out of order, it doesn't really matter because the client can figure out that the packets are out of order and make another request. The reliability of TCP is unnecessary in this instance because it causes performance degradation and may hinder the usefulness of the service.

Another example of a service that doesn't need the guarantee of a reliable channel is the ping command. The purpose of the ping command is to test the communication between two programs over the network. In fact, ping needs to know about dropped or out-of-order packets to determine how good or bad the connection is. A reliable channel would invalidate this service altogether.

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data from one application to another.

Note: Many firewalls and routers have been configured not to allow UDP packets. If you're having trouble connecting to a service outside your firewall, or if clients are having trouble connecting to your service, ask your system administrator if UDP is permitted.

Thus, choose UDP if:

- Reliability is not an issue. If the data is such that later packets contain information that supersedes that contained in earlier packets, then the information in a lost packet will become irrelevant when the next packet does actually arrive.

- UDP allows one packet to be broadcast (sent to everyone on the network), or multicast (addressed to a group of machines). This is done by addressing it to either the broadcast address (to which everyone listens), or a multicast address for the group (to which all the members of that group subscribe). This avoids the need of repeating the same information to a group of recipients by making copies of the information and sending a separate packet to every recipient.

C.1.1.3 Understanding Ports

Generally speaking, a computer has a single physical connection to the network. All data destined for a particular computer arrives through that connection. However, the data may be intended for different applications running on the computer. So how does the computer know to which application to forward the data? Through the use of ports.

Data transmitted over the Internet is accompanied by addressing information that identifies the computer and the port for which it is destined. The computer is identified by its 32-bit IP address, which it uses to deliver data to the right computer on the network. Ports are identified by a 16-bit number, which TCP and UDP use to deliver the data to the right application.

Service	Port
FTP (File transfer)	21
SSH (secure shell)	22
SMTP (email)	25
HTTP (web)	80

Table C.1: Port numbers for well known services.

In connection-based communication such as TCP, a server application binds itself to a specific port number. This has the effect of registering the server with the system to receive all data destined for that port. A client can then rendezvous with the server at the server's port.

Definition: The TCP and UDP protocols use ports to map incoming data to a particular process running on a computer.

In datagram-based communication such as UDP, the datagram packet contains the port number of its destination and UDP routes the packet to the appropriate application.

Port numbers range from 0 to 65535 because ports are represented by 16-bit numbers. The port numbers ranging from 0 - 1023 are restricted; they are reserved for use by well-known services such as HTTP and FTP and other system services. These ports are called well-known ports. Your applications should not attempt to bind to them.

Some of the port numbers corresponding to standard services are shown in Table C.1.

Clients will generally choose free port numbers in the range 1024-65535 for their side of the communication, just for the purpose of having a place to receive the replies. Their port number does not have to be standardized because it is part of the information transmitted to the server when the client first connects. Servers, on the other hand, have to use standard port numbers so that the client can know where to find them.

The host name/IP number and the port number together represent the postal address of a particular service (for example, a mail service may have the address mail.ru.ac.za:25, or 146.231.128.1:25 - since the protocols use the numeric form, the text version is just for human readers). Everyone, senders and receivers alike need an address of this kind. Servers will usually choose a particular port number, but even clients who send requests to the servers will need a port number. In the latter case, a free port number can be allocated automatically by the system's networking software. This is the port number that must be used if the server is to address a response back to the client.

C.1.1.4 Sockets

Communication between two machines involves both parties having registered an address at which they can receive packets. This address, in the case of Internet communication, involves the machine address (an IP number) and a port number (corresponding to the service).

This address is usually embodied in a software construct known as a socket. A socket is effectively a data structure storing among other things, an IP number and a port that is being used for communication by a networked application.

Protocols such as TCP establish (virtual) connections between two networked applications. A connection can be viewed as a pair of sockets. Each socket is an endpoint of the connection. Thus a socket can also be seen as the endpoint of a connection, or a place to connect to (hence the name).

C.1.2 Networking Classes in Java

Through the classes in java.net, Java programs can use TCP or UDP to communicate over the Internet. The URL, URLConnection, Socket, and ServerSocket classes all use TCP to communicate over the network. The DatagramPacket, DatagramSocket, and MulticastSocket classes are for use with UDP.

C.1.3 TCP Sockets

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

The client knows the host name of the machine on which the server is running and the port number to which the server is connected. To make a connection request, the client tries to rendezvous with the server on the server's machine and port.

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to a different port. It needs a new socket (and consequently a different port number) so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. Note that the socket on the client side is not bound to the port number used to rendezvous with the server. Rather, the client is assigned a port number local to the machine on which the client is running.

The client and server can now communicate by writing to or reading from their sockets.

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between your Java program and another program on the network. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the java.net.Socket class instead of relying on native code, your Java programs can communicate over the network in a platform-independent fashion.

C.1.3.1 Creating the connection

Complete program code for the server is given in Figure C.1, and for the client in Figure C.2.

Firstly, the server creates a special socket which listens for new connections on a given port (the port associated with the service, in this case 1235):

```
listenSocket = new ServerSocket (1235);
```

ServerSocket is a java.net class that provides a system-independent implementation of the server side of a client/server socket connection. The constructor for ServerSocket throws an exception if it can't listen on the specified port (for example, the port is already being used).

The server then waits for connections. The accept method blocks until a client tries to connect.

```
serverSocket = listenSocket.accept ();
```

At some point a client will want to connect to this service. The client creates its own socket (representing the other side of the connection) and specifies the location of the service.

```
socket = new Socket (serverhostname, 1235);
```

Back on the server side, the server detects the attempt at a connection. The server creates a copy of its socket, fills in extra details relating to the connection being made and returns the result when the accept call ends.

The server can now communicate with the client using this new socket. Unless the communication is going to be very brief, it is likely that the server will create a separate thread to do the communication, and loop around to repeat the accept call. This will allow it to handle further requests from other clients.

```
import java.io.*;
import java.net.*;
import java.util.*;
class TCPServer
  {
    public static void main (String args [])
      {
        ServerSocket listenSocket = null;
        try
          {
            // the server listens on port 1235.
            listenSocket = new ServerSocket (1235);
          }
        catch (SocketException e)
          {
            System.err.println ("Could not create server listening socket.");
            System.exit (1);
          }
        catch (IOException e)
          {
            System.err.println ("Error creating server listening socket.");
            System.exit (1);
          }

        // listen for a new connection. Since multiple connections
        // can be supported, each new connection is associated with
        // a new socket.
        Socket serverSocket = null;
        try
          {
            serverSocket = listenSocket.accept ();
          }
        catch (IOException e)
          {
            System.err.println ("Accept failed.");
          }
        // a new connection has been created.
        if (serverSocket != null)
          {
            // retrieve the message from the stream.
            try
              {
                InputStream inputStream = serverSocket.getInputStream ();
                int b = inputStream.read ();
                System.out.println ("First byte in received packet is " + b);
              }
            catch (IOException e)
              {
                System.err.println ("Error reading data from request.");
              }

            try
              {
                serverSocket.close ();
              }
            catch (IOException f)
              {
                System.err.println ("Error closing socket.");
              }
          }

        try
          {
            listenSocket.close();
          }
        catch (IOException e)
          {
            System.err.println ("Unable to close server listening socket.");
          }
      }
  }
```

Figure C.1: Sample TCP Server code.

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class TCPClient
  {
    public static void main (String[] args)
      {
        String hostname;
        if (args.length != 1)
          {
            System.out.println ("Require hostname of server machine as an argument.");
            return;
          }
        hostname = args[0];

        Socket socket = null;
        try
          {
            socket = new Socket (hostname, 1235);
          }
        catch (SocketException e)
          {
            System.err.println ("Client cannot create socket");
            return;
          }
        catch (UnknownHostException e)
          {
            System.err.println ("Client cannot find host: " + hostname);
            return;
          }
        catch (IOException e)
          {
            System.err.println ("Error when client creates socket.");
            return;
          }
        OutputStream outputStream = null;
        try
          {
            outputStream = socket.getOutputStream ();
            byte [] data = new byte [10];
            data[0] = (byte) 89;
            outputStream.write (data);
            outputStream.flush ();
          }
        catch (IOException e)
          {
            System.err.println ("Not able to create a outputstream to populate outgoing packet.");
          }
        try
          {
            socket.close();
          }
        catch (IOException e)
          {
            System.err.println ("Error when client closes socket.");
          }
      }
  }
```

Figure C.2: Sample TCP Client code.

C.1.3.2 Reading from and Writing to a Socket

Each socket supplies separates streams for transmission and reception.

```
outputStream = socket.getOutputStream();
```

All data communicated over a network ends up as sequences of octets (bytes). All data structures that need to be transmitted must eventually be converted to this format.

```
byte [] data = new byte [10];
data[0] = (byte) 89; // create some data!
outputStream.write(data);
outputStream.flush();
```

Reading objects uses a similar process:

```
InputStream inputStream = serverSocket.getInputStream();
int b = inputStream.read(); // read one byte.
System.out.println("First byte in received packet is " + b);
```

A well-behaved program always cleans up after itself. You should close any streams connected to a socket before you close the socket itself.

However, the basics of any client program are much the same:

1. Open a socket.
2. Open an input stream and output stream to the socket.
3. Read from and write to the stream according to the server's protocol.
4. Close the streams.
5. Close the socket.

Only step 3 differs from client to client, depending on the server. The other steps remain largely the same.

C.1.4 Datagram sockets

Clients and servers that communicate via a reliable channel, such as a URL or a socket, have a dedicated point-to-point channel between themselves, or at least the illusion of one. To communicate, they establish a connection, transmit the data, and then close the connection. All data sent over the channel is received in the same order in which it was sent. This is guaranteed by the channel.

In contrast, applications that communicate via datagrams send and receive completely independent packets of information. These clients and servers do not have and do not need a dedicated point-to-point channel. The delivery of datagrams to their destinations is not guaranteed. Nor is the order of their arrival.

The java.net package contains two classes to help you write Java programs that use datagrams to send and receive packets over the network: DatagramSocket, and MulticastSocket. An application can send and receive DatagramPackets through a DatagramSocket. In addition, DatagramPackets can be broadcast to multiple recipients all listening to a MulticastSocket.

C.1.4.1 Working with Datagrams

The listing for a sample datagram server application is given in Figure C.3, and the corresponding client in Figure C.4.

Datagrams are convenient when sending periodic updates or announcements, particularly when it is not crucial if some of these are lost.

A client does not have to worry about creating connections. Messages can be fired at the destination address for the server, irrespective of whether the server is running or not.

```
import java.io.*;
import java.net.*;
import java.util.*;
class UDPServer
  {
    public static void main (String args [])
      {
        DatagramSocket socket = null;
        try
          {
            // the server runs on port 1234
            socket = new DatagramSocket (1234);
          }
        catch (SocketException e)
          {
            System.err.println ("Could not create server socket");
            System.exit (1);
          }
        int packetDataLength = 1024;
        byte [] packetData = new byte [packetDataLength];

        // receive request
        DatagramPacket packet = new DatagramPacket (packetData, packetDataLength);
        try
          {
            socket.receive(packet);
          }
        catch (IOException e)
          {
            System.err.println ("Unable to receive request.");
          }
        System.out.println ("Received packet");

        byte [] data = packet.getData ();
        System.out.println ("First byte in received packet is " + (int) data[0]);
        socket.close();
      }
  }
```

Figure C.3: Sample UDP Server code.

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class UDPClient
  {
    public static void main (String[] args)
      {
        String hostname;
        if (args.length != 1)
          {
            System.out.println ("Require hostname of server machine as an argument.");
            return;
          }
        hostname = args[0];
        DatagramSocket socket = null;
        DatagramPacket packet = null;
        try
          {
            // create a socket, let the system choose the port number.
            socket = new DatagramSocket();
          }
        catch (SocketException e)
          {
            System.err.println ("Client cannot create socket");
            System.exit (1);
          }
        // send request
        InetAddress address = null;
        try
          {
            address = InetAddress.getByName(hostname);
          }
        catch (UnknownHostException e)
          {
            System.err.println ("Host: " + hostname + " is unknown");
            System.exit (1);
          }
        System.out.println ("Sending packet to " + address);
        byte [] bufout = new byte [10]; // choose size based on content to send.
        bufout[0] = (byte) 74;
        // address packet to server socket address.
        packet = new DatagramPacket(bufout, bufout.length, address, 1234);
        try
          {
            socket.send(packet);
          }
        catch (IOException e)
          {
            System.err.println ("Client cannot send on socket");
          }
        System.out.println ("Packet sent");
        socket.close ();
      }
  }
```

Figure C.4: Sample UDP client code.

A new socket is created by specifying the local port. This is the port that the socket will receive packets on. Servers usually specify this, so that the service is always associated with a known port number. Clients do not specify a port, and so the system software automatically assigns them a free port number. The IP address is taken automatically from the address of the machine upon which the program is running. Here a socket is created for a service on port 1234.

```
socket = new DatagramSocket (1234);
```

Each packet must be addressed independently. The addresses used can be converted from the text version ("delta.ru.ac.za" or "146.231.120.1") to the InetAddress structure as follows:

```
InetAddress address = null;
address = InetAddress.getByNam e (server);
```

As for the TCP case, data structures must be converted into byte arrays before they can be placed in packets. The packet can be created once the contents and destination address are available.

```
byte [] bufout = new byte [10];
  // choose size based on content to send.
bufout[0] = (byte) 74;
// address packet to server socket address.
packet = new DatagramPacket(bufou t, bufout.length, address, 1234);
DatagramPacket packet = new DatagramPacket(data , data.length,
                                           address, port);
```

Sending the packet requires only a single call:

```
socket.send (packet);
```

The server behaves in a similar manner to the client, except that usually the server waits to receive receive requests before sending responses. The server can extract the return address for the client from the request packet

C.2 Socket Programming in C

C.2.1 Socket Concepts

C.2.1.1 Host names and IP numbers:

Hosts have names (for example: julian.uwo.ca) but sockets use IP addressing, which is by number (such as: [129.100.2.12]). In the old days name/number translations were tabled in */etc/hosts*. These days name to number translations are implemented by the Domain Name Service (or DNS).

Programmers don't scan /etc/hosts nor do they communicate with the DNS. The C library routines *gethostbyname* and *gethostbyaddr* each return a pointer to an object with the following structure:

```
struct hostent
  {
    char *h_name; /* official name */
    char **h_aliases; /* alias list */
    int h_addrtype; /* address type */
    int h_length; /* address length */
    char **h_addr_list; /* address list */
  };
#define h_addr h_addr_list[0] /* backward compatibility */
```

The structure *h_addr_list* is a list of IP numbers (recall that a machine might have several interfaces, each will have a number). Good programmers would try to connect to each address listed in turn, lazy programmers just use *h_addr* - the first address listed. The connection is usually prefaced by translating a host name into an IP number.

```
struct hostent *hp;
if ((hp = gethostbyname(host)) == NULL) then
  error...
else
  int ipaddr = *(int *)(*(hp->h_addr_list ));
```

C.2.1.2 Services and Ports

Client applications connect to a host.port for a service provided by the application found at that address. Some services have names (eg. SMTP the Simple Mail Transfer Protocol). Ports have numbers (eg. SMTP is a service on port 25). The mapping from service names to port numbers is listed in */etc/services*, for the established services. But programmers don't scan /etc/services, they use library routines. The C library routines *getservbyname* and *getservbyport* each return a pointer to an object with the following structure containing the broken-out fields of a line in /etc/services.

```
struct servent
  {
    char *s_name; /* name of service */
    char **s_aliases; /* alias list */
    int s_port; /* port for service */
    char *s_proto; /* protocol to use */
  };
```

Server applications for new services will not yet have a registered port number, and must device their own. Port numbers between 0 and 1024 can only be used by system servers (those running with root permissions). Port numbers from 1025 to 65535 can be used by user level servers.

Client applications connect to a service port. Usually this is prefaced by translating a service name (eg. SMTP) into the port number but if you know the port number you can skip that step.

C.2.1.3 Byte ordering

Now that you can talk between machines, you have to be careful what you say. Many machines use differing dialects, such as ASCII versus EBCDIC. More commonly there are byte-order problems. Unless you always pass text, you'll run up against the byte-order problem. Luckily people have already figured out what to do about it.

Once upon a time in the dark ages someone decided which byte order was "right". Now there exist functions that convert one to the other if necessary. Some of these functions are *htons* (host to network short integer), *ntohs* (network to host short integer), *htonl* (host to network long integer), and *ntohl* (network to host long integer). Before sending an integer through a socket, you should first massage it with the *htonl* function:

```
i= htonl(i);
write_data(s, &i, sizeof(i));
```

and after reading data you should convert it back with *ntohl*:

```
read_data(s, &i, sizeof(i));
i= ntohl(i);
```

If you keep in the habit of using these functions you'll be less likely to goof it up in those circumstances where it is necessary.

C.2.1.4 Socket Addressing

A Socket Address is a host.port pair (communication is between host.port pairs – one on the server, the other on the client). We know how to determine host numbers and service numbers so we're well on our way to filling out a structure where we specify those numbers. The structure is sockaddr_in, which has the address family is AF_INET as in this fragment:

```
int port;
int ipaddr;
struct sockaddr_in cliadd;
memset (&cliadd, 0, sizeof (struct sockaddr_in));
cliadd.sin_family = AF_INET;
cliadd.sin_addr.s_add r = ipaddr;
cliadd.sin_port = htons (port);
```

C.2.1.5 Sockets

Sockets are just like "worm holes" in science fiction. When things go into one end, they (should) come out of the other. Different kinds of sockets have different properties. Sockets are either connection-oriented or connectionless. Connection-oriented sockets allow for data to flow back and forth as needed, while connectionless sockets (also known as datagram sockets) allow only one message at a time to be transmitted, without an open connection. There are also different socket families. The two most common are AF_INET for internet connections, and AF_UNIX for unix IPC (interprocess communication).

A socket is a Unix file descriptor created by the *socket* call.

```
int skt;
skt = socket(domain, type, protocol);
```

The *domain* parameter specifies a communications domain (or address family). For IP use AF_INET. The *type* parameter specifies the semantics of communication (sometimes know as a specification of quality of services). For TCP/IP use SOCK_STREAM, for UDP/IP use SOCK_DGRAM. A SOCK_STREAM is a sequenced, reliable, two-way connection based byte stream. If a data cannot be successfully transmitted within a reasonable length of time the connection is considered broken and I/O calls will indicate an error. The *protocol* specifies a particular protocol to be used with the socket, currently use 0.

C.2.1.6 Binding Sockets

A socket is created without a name. Until a name is bound to the socket, processes have no way to reference it, and, consequently, no messages can be received on it. The *bind* system call allows a process to specify half of an association, *local address: local port*.

```
if (bind (skt, (struct sockaddr *) &cliadd, sizeof (cliadd)) < 0)
  {
    cerr << "Error trying to bind\n";
    close (skt);
    exit (1);
  }
```

The IP number of the local machine can be set as INADDR_ANY to have the system use the local machine name. Port can be set to 0 if you wish the system to find and use a free port number (useful for clients).

C.2.1.7 Discarding Sockets

A socket is discarded by closing the descriptor; use the *close* system call:

```
close(s);
```

If data is associated with a socket that promises reliable delivery (for example, a stream socket) when a close takes place, the system will continue trying to transfer the data. However, after a period of time, undelivered data is discarded.

C.2.2 Datagram Sockets

Datagram sockets are created as described in "Creating Sockets". If a particular local address is needed, the bind() operation must precede the first data transmission. Otherwise, the system will set the local address and/or port when data is first sent.

To send datagrams, one must be allowed to specify the destination. The call *sendto* takes a destination address as an argument and is therefore used for sending datagrams. The call *recvfrom* is often used to read datagrams, since this call returns the address of the sender, if it is available, along with the data. If the identity of the sender does not matter, one may use *recv*.

C.2.2.1 Connectionless Servers

While connection-based services are the norm, some services are based on the use of datagram sockets. The rwho service is an example. It provides users with status information for hosts connected to a local area network. This service is predicated on the ability to broadcast information to all hosts connected to a particular network.

A user on any machine running the rwho server can find out the current status of a machine with the ruptime program.

The rwho server, in a simplified form, is shown in this code sample:

```
main()
  {
    ...
    sp = getservbyname("who", "udp");
    from.sin_addr.s_addr = htonl(INADDR_ANY);
    from.sin_port = sp->s_port;
    ...
    s = socket(AF_INET, SOCK_DGRAM, 0);
    ...
    bind(s, (struct sockaddr *)&from, sizeof (from));
    ...
    for (;;)
      {
        struct whod wd;
        int cc, whod, len = sizeof (from);
        cc = recvfrom(s, (char *)&wd, sizeof (struct whod), 0,
           (struct sockaddr *)&from, &len);
        if (cc <= 0)
          {
            syslog(LOG_ERR, "rwhod: recv: %m");
            continue;
          }
        if (!verify(wd.wd_hostname))
          {
            syslog(LOG_ERR, "rwhod: malformed host name from %x",
              ntohl(from.sin_addr.s_addr));
            continue;
          }
        wd.wd_sendtime = ntohl(wd.wd_sendtime);
        ...
```

```
        sendto (s, (char *)&wd, sizeof (struct whod), 0,
          (struct sockaddr *) &from, sizeof (from));
```

C.2.2.2 Connectionless Clients

An example of a client for a connectionless server is shown below:

```
sockid = socket(AF_INET, SOCK_DGRAM, 0);
bzero((char *) &my_addr, sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
my_addr.sin_port = htons(0);
if ((bind (sockid, (struct sockaddr *) &my_addr, sizeof(my_addr)) < 0))
  {
    printf("Client: bind fail: %d\n",errno);
    exit(0);
  }
bzero((char *) &server_addr, sizeof(server_addr ));
server_addr.sin_family = AF_INET;
server_addr.sin_addr. s_addr = inet_addr(SERV_HOST_ ADDR);
server_addr.sin_port = htons(SERVER_PORT_ ID) ;
retcode = sendto(sockid, msg, 12, 0,(struct sockaddr *) &server_addr,
  sizeof(server_addr) );
socklen_t server_addr_len;
server_addr_len = sizeof (server_addr);
length = recvfrom (sockid, message, MAXDGRAM, 0,
                   (struct sockaddr *) &server_addr, &server_addr_len);
close(sockid);
```

C.2.2.3 Nonblocking Sockets

Programs that cannot wait for a socket operation to be completed should use nonblocking sockets. I/O requests on nonblocking sockets return with an error if the request cannot be satisfied immediately.

Once a socket has been created with the *socket* call, it can be marked as nonblocking by *fcntl* as follows:

```
s = socket(AF_INET, SOCK_STREAM, 0);
...
if (fcntl(s, F_SETFL, FNDELAY) < 0)
  {
    perror("fcntl F_SETFL, FNDELAY"); exit(1);
  }
...
```

When performing nonblocking I/O on sockets, check for the error EAGAIN (stored in the global variable errno). This error occurs when an operation would normally block, but the socket it was performed on is nonblocking. In particular, *accept*, *connect*, *send*, *recv*, *read*, and *write* can all return EAGAIN, and processes should be prepared to deal with this return code.

Bibliography

[1] *Hardware Choices and Explanations,* Debra Fligor, available via the WWW at http://choices.cs.uiuc.edu/schools/dfl-choice.html.

[2] *Ethernet,* H. Gilbert, available via the WWW at http://pclt.cis.yale.edu/pclt/comm/ether.htm.

[3] *Quick Reference Guides to 100 Mbps Fast Ethernet,* Charles Spurgeon, available via the WWW at http://wwwhost.cts.utexas.edu/ethernet/descript-100quickref.html.

[4] *Basic Glossary on Campus Networks,* John Wobus, available via the WWW at http://web.syr.edu/~jmwobus/lans/comfaqs/lan-glossary.html.

[5] *FAQ: Ethernet frame formats,* James Harvey, available via the WWW at http://web.syr.edu/~jmwobus/lans/comfaqs/faq-ethernet-format.

[6] *Ethernet Network Questions and Answers,* available via the WWW at http://wwwhost.cts.utexas.edu/ethernet/enet-faqs/ethernet-faq.

[7] *TechEncyclopedia,* available via the WWW as http://www.techweb.com/encyclopedia/.

[8] *Dictionary of PC Hardware and Data Communication Terms*, Mitchell Shnier, available via the WWW at http://www.oreilly.com/reference/dictionary/.

[9] *Introduction to Cryptography,* Tatu Ylonen, available via the WWW at http://www.cs.hut.fi/ssh/crypto/intro.html.

[10] *Cryptographic Algorithms,* Tatu Ylonen, available via the WWW at http://www.cs.hut.fi/ssh/crypto/algorithms.html.

[11] *Answers To FREQUENTLY ASKED QUESTIONS About Today's Cryptography,* Paul Fahn, available via the WWW at ftp://ftp.funet.fi/pub/crypt/cryptography/asymmetric/rsa.

[12] *Handbook of Applied Cryptography - Chapter 7,* Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, available via the WWW at http://www.cms.auburn.edu/hac/.

[13] *SNMP - Simple Network Management Protocol,* Yoram Cohen, available via the WWW at http://www.rad.com/networks/1995/snmp/snmp.htm.

[14] *WWW Security,* Yogev Mashiach and Ron Mertens, available via the WWW at http://www.rad.com/networks/1997/security/index.htm.

[15] *Suggestions for Random Number Generation in Software,* Tim Matthews, available via the WWW at http://www.rsa.com/rsa/developers/randomn.htm.

[16] *comp.dcom.lans.token-ring Frequently Asked Questions,* James Messer, available via the WWW at http://home.sprynet.com/sprynet/jtmesser/faq/contents.html.

[17] *Internet Protocol Frequently Asked Questions,* George V. Neville-Neil, available via the WWW at http://netman.cit.buffalo.edu/FAQs/top-ip.faq.

[18] *comp.dcom.isdn FAQ,* Dave Cherkus, available via the WW at ftp://rtfm.mit.edu/pub/usenet/news.answers/isdn-faq/.

[19] *Networks for Dummies,* Gil Behar, Guy Loewy and Oz Solomonovich, available via the WWW at http://www.rad.com/networks/1997/nettut/mainmenu.html.

[20] *FDDI- Fiber Distributed Data Interface,* Chen Frenkel and Tanya Abarbanel, available via the WWW at http://www.rad.com/networks/1995/fddi/fddi.htm.

[21] *ATM tutorial,* available via the WWW at http://www.rad.com/networks/1994/atm/tutorial.htm.

[22] *Designing Switched Networks,* available via the WWW at http://www.hp.com/rnd/techlib/ethernet_switching/design.htm.

[23] *Introduction to Petri Nets,* Arash Khodabandeh, available via the WWW at http://rd13doc.cern.ch/public/doc/Note54/Note54.html.

[24] *Short Introduction to Petri Nets,* Miguel Menasche, available via the WWW at http://www.ele.puc-rio.br/~menasche/petri/petridef.html..

[25] *Approaches to Performance Evaluation,* Wan Ling Li, available via the WWW at http://outoften.doc.ic.ac.uk/~nd/surprise_97/journal/vol2/wll1/queues.html.

[26] *Various Kinds of Petri Nets in Simulation and Modelling,* Tomas Vojnar, available via the WWW at http://www.fee.vutbr.cz/UIVT/homes/vojnar/Articles/mosis97/mosis97.html.cz.

[27] *Firewall design,* D. Brent Chapman and Elizabeth D. Zwicky, available via the WWW at http://www.sunworldonline/swol-01-1996/swol-01-firewall.html.

[28] *Petri Nets with time parameters,* Miguel Menasche, available via the WWW at http://www.ele.puc-rio.br/~menasche/petri/time/.

[29] *A Petri Net Design, Simulation, and Verification Tool,* Richard Scott Brink, available via the WWW at http://www.csh.rit.edu/~rick/thesis/doc/PetriThesis.html.

[30] *A comparative analysis of the Security of NT versus UNIX,* Patrick Kennedy, available via the WWW at http://www.csn.ul.ie/~duke/os.html.

[31] *Stochastic Models in POM,* Alan Scheller-Wolf, available via the WWW at http://mat.gsia.cmu.edu/STOCHASTICS.

[32] *TCP-IP Networks,* Alex Peeters, available via the WWW at http://www.citap.com/documents/tcp-ip/tcpip.htm.

[33] *TCP/IP and IPX routing Tutorial*, Sangoma Technologies, available via the WWW at http://www.sangoma.com/fguide.htm.

[34] *An Overview of TCP/IP Protocols and the Internet,* Gary C. Kessler, available via the WWW at http://www.hill.com/library/tcpip.html.

[35] *RFC 791: Internet Protocol: DARPA Internet Program Protocol Specification,* available via the WWW at ftp://ftp.isi.edu/in-notes/rfc791.txt.

[36] *Network Standards,* Lee James McMunn, available via the WWW at http://www.awsteven.son.demon.co.uk/SYSNOTES/iso.htm.

[37] *RFC792: Internet Control Message Protocol*: *DARPA Internet Program Protocol Specificaction*, available via the WWW at ftp://ftp.isi.edu/in-notes/rfc792.txt.

[38] *RFC 793: Transmission Control Protocol: DARPA Internet Program Protocol Specificaction*, available via the WWW at ftp://ftp.isi.edu/in-notes/rfc793.txt.

[39] *Switching: Is a Switched Ethernet Network Right For You?*, available via the WWW at http://www.dayna.com/dayna/solutions/tutorials/switchtut.html.

[40] *Network Management,* Yogesh Agarwal, available via the WWW at http://www.geociti es.com/SiliconValley/Peaks/9363/snmp1.htm.

[41] *Network Management Basics,* Cisco Systems, available via the WWW at http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/55018.htm

[42] *An Introduction to Network Management,* Tyler Kvallil, available via the WWW at http://www.inforamp.net/~kjvallil/t/snmp.html.

[43] *Managing a network architecture: Insights to managing your network,* Harris Kern, Randy Johnson, Michael Hawkins, Andrew Law and William Kennedy, available via the WWW at http://www.sunworld.com/swol-02-1996/swol-02-hrbook.html.

[44] Lan/Network Acronyms, available via the WWW at http://atzhim3.gordon.army.mil/css/training/LAN/ACRONYMS.htm.

[45] *DAICI (Diccionario de Acronimos en Ingenieria, Comunicaciones e Informatica)*, Juan Carlos Miguez, available via the WWW at http://www.cs.columbia.edu/~hgs/acronyms.

[46] *An Introduction to SNMP,* Anixter technical library, available via the WWW at http://www.anixter.com/techlib/buying/network/snmp1.htm.

[47] *The Basics of Networking,* Asante, available via the WWW at http://www.asante.com/education_primer/page3.html.

[48] Computer Networks, S. A. Trainis, available via the WWW at http://www.cs.herts.ac.uk/~simon/networks1/networks1.html.

[49] *Routing,* Carl Erickson, available via the WWW at http://www.dbcs.uu.se/~carle/datakomm/Notes/Networking/31_RoutingAlgorithms.html.

[50] *Configuring Bridging Services,* Bay Networks, available via the WWW at http://support.baynetworks.com/library/tpubs/html/router/soft1000/bridge/2950A-1.html.

[51] *Communications Networks,* Sarit Mukherjee, available via the WWW at http://www.nclab.hanyang.ac.kr/Resource/Seminar/network/courses.html.

[52] *The Minimum Spanning Tree Problem,* available via the WWW at http://ftp.orie.cornell.edu/~or115/handouts/handout4/handout4.html.

[53] *Ethernet Backoff Algorithm*, Hui Dang, available via the WWW at http://bugs.wpi.edu:8080/EE535/hwk96/hwk3cd96/dang/dang.html.

[54] *Introduction to Computational Statistical Mechanics*, Vilia Payne, available via the WWW at http://www.molres.org/vilia/lectures/lectures.html.

[55] *Extensible Hierarchical Object-Oriented Logic Simulation with an Adaptable Graphical User Interface*, Donald C. Craig, available on the WWW at http://www.cs.mun.ca/~donald/msc/thesis.html.

[56] *Formal Verification and Empirical Analysis of Rollback Relaxation*, Kothanda Umamageswaran, Krishnan Subramani, Philip A. Wilsey and Perry Alexander, available via the WWW at http://www.ececs.uc.edu/~koduri/papers/jsa97/html/main.html.

[57] *Telecommunications: Glossary of Telecommunication Terms*, available via the WWW at http://www.its.bldrdoc.gov/fs-1037/.

[58] *Network Essentials*, available via the WWW at http://www.bestweb.net/~omalley/NetEssentials/.

[59] *Introduction to Networks*, Lim Pei Mun J, available via the WWW at http://www.geociti es.com/SiliconValley/Horizon/6488/articles/netintro.htm.

[60] *CEN 4500C Fundamentals of Computer Communication Networks*, Richard Newman, available via the WWW at http://www.cise.ufl.edu/~nemo/cen4500/index.html.

[61] *Encoding (NRZ, NRZI, Manchester, 4B/5B)*, Morgan Kaufman Publishers, Inc., available via the WWW at http://www.mkp.com/books_catalog/cn/book/node27.htm.

[62] *EG 3561 Communications Engineering*, Gorry Fairhurst, available via the WWW at http://www.erg.abdn.ac.uk/users/gorry/eg3561/lecturer.html.

[63] *A Painless Guide to CRC Error Detection Algorithms*, Ross N. Williams, available via the WWW at ftp://coast.cs.purdue.edu/pub/doc/authentication/painless-guide-to-crc.text.Z.

[64] *Primer on Fiber Optic Data Communications for the Premises Environment*, Kenneth S. Schneider, available via the WWW at http://telebyteusa.com/fqprimer/fofull.htm.

[65] *Section I: Lecture Topics, Key Terms, and Review Question Solutions*, Daniel Morrow, available via the WWW at http://voyager.elec.sthasta.cc.ca.us/~dmorrow/ftp_files/SECTL.HTML.

[66] *Introduction to Simulation*, M. A. Pollatschek, available via the WWW at http://iew3.technion.ac.il:8080/~bani/simint.html.

[67] *An Introduction to Simulation*, Michael A. Trick, available via the WWW at http://mat.gsia.cmu.edu/simul/simul.html.

[68] *RFC1157 - A Simple Network Management Protocol (SNMP)*, J. Case, M. Schoffstall, J. Davin, available via the WWW at http://www.cis.ohio-state.edu/htbin/rfc/rfc1157.html.

[69] *CS4514 Computer Networks*, Craig E. Wills, available via the WWW at http://www.cs.wpi.edu/~cew/.

[70] *CS363 - Lecture Notes*, Ralph Droms, available via the WWW at http://info.fedu.metu.edu.tr/~hasan/www/314/ttt/cs363/lecture_notes/lecture_notes.html.

[71] *Computer Networking*, P J Willis, available via the WWW at http://www.maths.bath.ac.uk/~pjw/NOTES/networks/networks.html.

[72] *A Layman's Guide to a Subset of ASN.1, BER, and DER*, Burton S. Kaliski Jr., available via the WWW at ftp://ftp.rsasecurity.com/pub/pkcs/ascii/layman.asc.

[73] *Introduction to Socket Programming*, Reg Quinton, available via the WWW at http://www.uwo.ca/its/doc/courses/notes/socket/index.html.

[74] *BSD Sockets: A Quick And Dirty Primer*, Jim Frost, available via the WWW at http://ftp.std.com/homepages/jimf/sockets.html.

[75] *Unix-socket-faq for network programming*, Vic Metcalfe, available via the WWW at http://www.faqs.org/faqs/unix-faq/socket/index.html.

[76] *IRIX Network Programming Guide*, Susan Thomas, Jed Hartman, Judith Radin, Helen Vanderberg and Terry Schultz, available via the WWW at http://ask.ii.uib.no/ebt-bin/nph-dweb/dynaweb/SGI_Developer/IRIX_NetPG/@Generic__BookTextView/4;cs=full.html;pt=55.

[77] *The Java Tutorial: A practical guide for programmers*, Lisa Friendly, Mary Campione, Kathy Walrath, Alison Huml, available via the WWW at http://java.sun.com /docs/books/tutorial/index.html .

[78] *Introduction to Spread Spectrum*, Randy Roberts, available via the WWW at http://www.sss-mag.com/ss.html#tutorial .

[79] *How LAN Switches Work*, Cisco Systems, Inc, available via the WWW at http://www.cisco.com /warp/public/473/lan-switch-cisco.shtml.

[80] *Gigabit Ethernet*, Vijay Moorthy, available via the WWW at ftp://ftp.netlab.ohio-state.edu/pub/jain/courses/cis788-97/gigabit_ethernet/index.htm

[81] *Wireless Local Area Networks*, Edward C. Prem, available via the WWW at ftp://ftp.netlab.ohio-state.edu/pub/jain/courses/cis788-97/wireless_lans/index.htm.

[82] *Tutorial - Lesson 137: Wireless LANs*, Jonathan Angel, available via the WWW at http://www.networkmagazine.com/article/NMG20000517S0164 .

[83] *Introduction to IEEE 802.11*, Intelligraphics Inc., available via the WWW at http://www.intelligraphics.com/articles/802.11_article.html.

[84] *802.11 MAC Layer Defined*, Jim Geier, available via the WWW at http://www.80211-planet.com/tutorials/article/0,4000,10724_1216351,00.html.

[85] *SNMPv3: A Security Enhancement for SNMP*, William Stallings, available via the WWW at http://www.comsoc.org/livepubs/surveys/public/4q98issue/stallings.html.

[86] *CERT Incidents and Advisories*, available via the WWW at http://www.cert.org .

[87] *RFC 2026: The Internet Standards Process – Revision 3*, Scott O. Bradner, available via the WWW at http://www.ietf.org/rfc/rfc2026.txt.

[88] *The Internet Engineering Task Force*, available via the WWW at http://www.ietf.org/ .

[89] *Digital Subscriber Line*, Cisco Systems, available via the WWW at http://www.cisco.com /univercd/cc/td/doc/cisintwk/ito_doc/adsl.htm.

[90] *comp.dcom.xdsl Frequently Asked Questions*, John Kristoff, available via the WWW at http://homepage.interaccess.com/~jkristof/xdsl-faq.txt.

[91] *Asymmetric Digital Subscriber Line*, Eyal Ayalon, Tzachi Levy, Guy Kerer, available via the WWW at http://www2.rad.com/networks/1997/adsl/AdslMainPage.htm.