High Speed Networks
Study Guide

Dr. Wanlei Zhou
and
Dr. Weijia Jia

Contents

0. Unit Outline

0.1. Objective

The objective of this unit is to provide an up-to-date knowledge of high-speed networks to students. At the end of this unit, students should be familiar with the basic concepts, the architectures, the protocols, the advantages and limitations, and the recent development of various high-speed networking technologies. The knowledge gained from the study of this unit will enable students to make decisions in selecting suitable high-speed networking technology for their own environment and applications. Students will also have adequate knowledge in tracking the current and future development in high-speed networks.

The Study Guide is divided into four parts. The first part (Sessions 1 and 2) surveys the development in high-speed networks. The second part (Sessions 3 and 4) deals with high-speed local networks. The third part (Sessions 5, 6, 7, 8, and 9) introduces high-speed internetworking techniques and QoS related issues. The fourth part (Sessions 10, and 11) addresses the issues of applications and network security. The Study Guide will concentrate on the following central problems that confront the designers and application developers of high-speed networks: the need to support multimedia and real-time traffic and applications, the need to control congestion, and the need to provide different levels of Quality of Services (QoS) to different applications.

0.2. Text Books

0.2.1. Main Text Book

"High-Speed Networks: TCP/IP and ATM Design Principles," By W. Stallings, Prentice-Hall, NJ, 1998. ISBN: 0-13-525965-7 ([STA98]).

0.2.2. Reference Books

- "FDDI – A High Speed Network" A. Shah and G. Ramakrishnan, Prentice-Hall, NJ, 1994. ISBN: 0-13-308388-8.
- "High-Speed Networking for Multimedia Applications," By W. Effelsberg, et al. Kluwer Academic Pub. 1996, ISBN: 0-7923-9681-2.
- "Handbook of Data Communications and Networks," By W. Buchanan, Kluwer Academic Pub. 1999.

0.3. Assessment

- Progressive assessment 40%
- Examination 60%

1. Introduction: The Need for Speed and Quality of Service

1.1. Study Points

- Major historical events of computer networks.
- Features of different high-speed networks.
- Requirement for high-speed networks.
- ATM Architecture and Protocols.

Reference: [STA98] Chapter 1.

1.2. A Brief Networking History

High-Speed Networks now dominate both wide-area network (WAN) and local-area network (LAN) markets. Public and private data networks have evolved from packet-switching network in 10s and 100s of kilo-bits per second (kbps), to frame relay network operating at up to 2 mega-bits per second (Mbps), and now Asynchronous Transfer Mode (ATM) operating at 155 Mbps or more. Fast Ethernet has evolved from 10-Mbps to 100-Mbps and now Gigabit Ethernet.

The following table ([STA98] p.3) shows a brief networking history.

1966	ARPA packet-switching experimentation
1969	First Arpanet nodes operational
1972	Distributed e-mail invented
1973	For non-U.S. computer linked to Arpanet
1975	Arpanet transitioned to Defense Communications Agency
1980	TCP/IP experimentation began
1981	New host added every 20 days
1983	TCP/IP switchover completed
1986	NSFnet backbone created
1990	Arpanet retired
1991	Gopher introduced
1991	WWW invented
1992	Mosaic introduced
1995	Internet backbone privatized
1996	OC-3 (155 Mbps) backbone built

1.3. Milestones

- The World Wide Web (WWW): In spring 1989, CERN (the European Lab. For Particle Physics), Tim B. Lee proposed the idea of a distributed hypermedia technology to facilitate international exchange of researches findings using Internet. In 1991, a prototype WWW called *line-oriented browser* was developed at CERN and released to limited population. Later, the *graphically oriented browser,* Mosaic, developed at the NCSA at University of Illinois by Mark Andresson and others in 1992. 2 million copies of Mosaic have been delivered in a short period.
- Email triggered rapid growth of ARPAnet so the Web triggered explosive growth in Internet.
- ISDN and Frame Relay: ISDN includes the integrated services and digital networking. The most noteworthy technical achievement of the ISDN effort was the development of specifications for Frame Relay. Frame Relay is now considered a matured technology. It has a proven track record and is used in many Fortune 500 companies. Banks are big users of Frame Relay services, especially their automated teller machine networks.
- B-ISDN: ISDN was designed to provide data rates in the range of Kbps to Mbps. To provide higher data rate—Mbps to Gbps—the original ISDN (also called Narrowband ISDN, or N-ISDN) was extended to Broadband ISDN (B-ISDN).
- ATM: Just as frame relay is a technology developed as part of the ISDN effort and now widely used in non-ISDN applications, so ATM is a technology developed as part of the B-ISDN effort and now widely used in non-B-ISDN applications. ATM is becoming popular because it supports transport of data, video and voice communications over a single infrastructure. While data is somewhat resilient, video and voice applications demand timely and sometimes high bandwidth. ATM brings together these features and offers high-speed communication. Most of today's LAN operate at 4Mbps or 10Mbps. High speed LANs can operate at 100Mbps. ATM provides LAN and WAN connectivity at 155Mbps. ATM standards are being developed in the marketplace. The cost of ATM has also started to decrease. With standardization and a better cost/benefit model, ATM should be one of the technologies of the future.
- Marketplace Examples: T-1 networking will continue to be a viable option for many years. While T-1's is not new technology, it is a proven high bandwidth solution for many applications, including imaging, data communication and video conferencing. Many government agencies and commercial companies have become big users of T-1's. It is safe to say that most states have successfully deployed T-1's as part of their communications infrastructure.
- ATM utilization is not as widely deployed as T-1 and Frame Relay communications. The military has begun implementation of ATM based projects. The US Army is deploying this technology at Fort Bragg, North Carolina, Fort Hood, Texas and Fort Stewart, Georgia. Three defense agencies at Fort Belvoir, Virginia are also using ATM as their backbone. Case Western Reserve University in Cleveland, Ohio is using ATM technology all the way to the desktop so students can work on

assignments and even take classes via video over the network. North Carolina has implemented an ATM backbone for education and government use across the state.

- Leading Vendors & Users: The regional telephone companies like Ameritech, BellSouth, Pacific Bell, etc., along with national communications providers like AT&T, MCI and US West are the major providers of both T-1 and Frame Relay networking services. ATM is still fairly new territory for vendors. There are a large number of hardware and software vendors providing ATM products today. Since there is not a clear standard for ATM, the major internetworking vendors seem to be the biggest players in this technology. Companies like Bay Networks, Cascade, Cisco, Digital Equipment Corporation, FORE Systems, General DataComm, Hewlett-Packard and IBM are just a few of the companies offering ATM products. Several telephone companies like MCI and US West are offering ATM solutions as well. Local and regional cable television providers have moved into this service area
- Public network infrastructure corresponds to B-ISDN and consists of public telecommunications networks that support public telephony, cable TV (CATV), and WAN services.
- ATM LAN: ATM switches can serve as a backbone network in a local setting to interconnect a variety of traditional LANs (e.g. Ethernet).
- ATM WAN includes enterprise networks operating over leased lines such as optical fiber and wireless links.

1.4. The Need for High Speed and Quality of Service (QoS)

Emergence of High-speed LANs: In the 1990s, 2 significant trends have altered the role of the PC and LAN: (1) The speed and computing power improvement of PCs, and (2) MIS (management information systems). Organisations have recognised the LAN as a viable and essential computing platform.

The following are examples of requirements that call for high-speed LANs:

- Centralized server farms: There is need for users, client systems to be able to access huge amounts of data from multiple centralized servers.
- Power work groups: These groups typically consist of a small number of cooperating users who need to draw massive data files across the network.
- High-speed local backbone.

Users of high-speed networks require different quality presentations at different times. These different quality presentations can be mapped into different parameter values. Quality of Service (QoS) specifies a set of parameters that can be assigned numerical values. However, quality of the presentation is a matter of user’s perception. This perceptual nature of QoS makes it subjective and difficult to quantify.

1.5. Advanced TCP/IP and ATM Networks

- Rely on enhanced IP-based Internets to prioritise time-sensitive traffic and provide enough capacity for all.

- ATM will be the network equivalent to provide high-speed service that is scalable to virtually and data rate.

1.5.1. IP-based Internet

- TCP/IP is the original protocol suite for internetworking. IP is a protocol for internet routing and delivery and TCP is a protocol for reliable end-to-end transport.
- Dynamic routing performs two vital functions: (1) Dynamic route discovery; end systems and routers do not have to be preconfigured to meet various possibilities or reconfigured with each change of topology. (2) It allows routers adjustment in the face of congestion or failure, resulting in efficient load balancing.
- Packet switching remains dominate technology
- Ethernet: from 10 Mbps → 100 Mbps or Gigabit Ethernet
- Emerge of ISA (Integrated Service Architecture) is to support multimedia and real-time applications, which involves upgrading router hardware to support real-time traffic and also a number of new protocols:
  - IPv6 – Next generation Internet protocol.
  - RSVP – The Resource reservation protocol enables users to reserve capacity on Internet and to specify the requirements of the traffic, in terms of desired throughput and delay characteristics.
  - RTP – The Real-time Transport Protocol provides mechanisms for delivering video, audio and other real-time traffic over Internet.
  - Multicast routing protocols: MOSPF, CBT, PIM etc.

1.5.2. ATM Networks

The key technical ingredients for successful ATM networking and now in place are:

- Internal signaling: Signaling System Number 7 (SS7) provides a set of protocols by which an ATM-based network can be managed effectively.
- External signaling: The user network interface for ATM networks includes mechanisms for setting up and tearing down connections.
- Physical layer specification: A powerful physical layer transmission specification is needed to support the high speed transmission of ATM. The mapping from ATM layer data transfer to the SDH (synchronous data hierarchy) / SONET (synchronous optical network) layer has been developed.
- ATM adaptation: The key to the support of the upper-layer protocols by ATM is the ATM adaptation layer (AAL) that maps various upper protocols and services onto the ATM layer.

1.6. Review Questions

1. Briefly describe the relationship of Frame Relay and ISDN.

2. Briefly describe the relationship of ATM and B-ISDN.
3. Why high-speed networking is so important? Use your own examples to illustrate the importance of high-speed networks.
4. What is QoS and why is it so important?

2. Data Networks and Communication Protocols

2.1. Study Points

- Familiar with various types of data networks
  1. Packet-switch networks
  2. X.25
  3. Frame Relay Networks
  4. Congestion and Traffic management
- Familiar with various types of protocol architectures
  1. OSI Reference Model
  2. X.25 Networks
  3. Internet and TCP/IP

Reference: [STA98] Chapter 3 and 2.

2.2. Packet-Switch Networks

2.2.1. Basic Concepts

Packet switching networks originated in 1970s and they are still widely used in today's wide-area networks. A *packet switching network* consists of a collection of interconnected packet-switching nodes (routers).

Packets: blocks of data with a typical length of 1000 octets (bytes). Each packet contains a *header* (control information) and a *data* portion.

Packet switching: data from the source are broken (disassembled) into packets and then routed over the network. At each node, the packet is received, stored briefly and passed on to the next node according to the header information and other criteria, such as traffic congestion, error conditions, the shortest end-to-end path, etc, until it reaches to the destination. At the destination, packets are assembled as a logical message for the presentment to the end user.

Advantages of packet switching:

- The alternative routing scheme improves the reliability of the network. Failed routers can be bypassed.
- Special packet switching computers and algorithms can be designed to minimise delay between end users. For example, priorities and data rate conversions can be implemented.
- The technology can use the existing telephone network for the path, therefore provided a value-added service for the end user.
- Packets containing a portion of user data are more secure than a complete message.

2.2.2. Switching and Routing Techniques

Two approaches are used in contemporary networks to handle streams of packets sent by end users, routing them through the network and delivering them to destinations.

- *Datagram*: in this approach each packet is treated independently. Each packet carries the destination address in its header. Therefore packets of the same message may be routed through a different path and they may arrive out of sequence at the exit node, or some of the packets may be lost. The reordering of packets and the recovery of lost packets can be carried out by the end user or by the exit node.
- *Virtual circuit*: in this approach a preplanned route is established between a pair of communicating parties before any packets are sent. Once the route is established, all packets between the pair follow the same route through the network. Each packet carries a virtual circuit identifier in its header, which is used by nodes of the virtual circuit to direct the packet along the virtual circuit.

The difference of the two approaches is that, the datagram approach needs to make a routing decision for every packet, whereas the virtual circuit approach only makes the routing decision once in the beginning.

More comparison:

- Virtual circuit may provide sequencing and error control services (such as recovery of lost packets).
- Packets may transit the network more rapidly in a virtual circuit since there is no need to make routing decisions. However, the initial call set up time is required.
- Datagram approach may be more efficient if only one or a few packets are to be transmitted since there is no call set up time.
- Datagram approach is more flexible since the routing algorithms can be used to adapt various situations.
- Datagram approach is inherently more reliable since it can bypass failed nodes.

Routing is an essential part of a packet switching network. The principal conditions that influence routing decisions are failures and congestion. Adaptive routing is a technique that changes the routing decisions as the network conditions change. However, adaptive routing algorithms rely on information exchanged among nodes, which may degrade the performance since the overhead involved in information exchanging and decision making.

2.2.3. X.25

X.25 is an ITU-T (International Telecommunication Union Telecommunication Standardisation Sector) standard that specifies an interface between a host system and a packet switching network.

X.25 consists of three levels:

- Physical level. It deals with the physical interface between an attached station (computer, terminal) and the link that attaches that station to the packet switching network. Physical level uses X.21 as well as other standards.
- Link level. It provides for the reliable transfer of data across the physical link by transmitting the data as a sequence of frames. The link level standard is referred to as LAPB (Link Access Protocol – Balanced).
- Packet level. It provides a virtual circuit service that enables any subscribers to the network to set up logical connections (virtual circuits) to other subscribers.

Figure 2.1 depicts the user data and X.25 protocol control information. User data are passed down to the X.25 packet level, which appends a header, creating a packet. The packet is then passed down to the link level, which appends a header and a trailer, forming a LAPB frame.

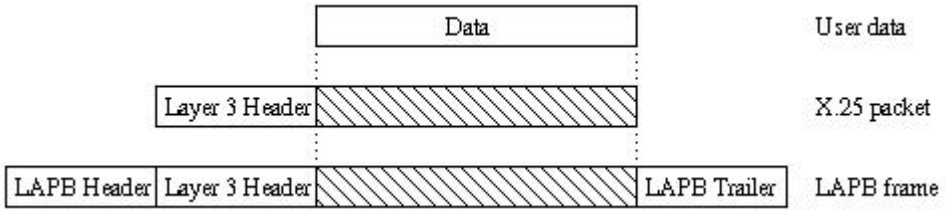


Figure 2.1 User data and X.25 protocol control information

Key features of X.25 protocol:

- Call control packets, used for setting up and clearing virtual circuits, are carried on the same channel and same virtual circuit as data packets.
- Multiplexing of virtual circuits takes place at layer 3.
- Both layer 2 and layer 3 include flow control and error control mechanisms.

2.3. Frame Relay Networks

2.3.1. Why Frame Relay?

- Overhead of X.25 approach associated with each node:
- Store data and organise them as a sequence of blocks.
- Add an X.25 header to each block to form a packet.
- Routing calculation.
- Form a LAPB frame from a frame by adding a LAPB header and a trailer.
- Transmit the frame to next node.
- Perform flow and error control functions when receiving a frame.
- Remove the data link layer fields and examine the packet header for routing purposes.

This overhead is not justified in today’s reliable and high-quality networks.

Frame relay is designed to eliminate much of the overhead of X.25. The key difference between frame relay and a conventional X.25 packet-switching network are:

- Call control signaling is carried on a separate logical connection from user data.
- Multiplexing and switching of logical connection take place at layer 2 instead of layer 3.
- There is no hop-by-hop flow control and error control.

Frame relay can result in lower delay and higher throughput.

2.3.2. Frame Relay Architecture

Figure 2.2 shows the protocol stacks of frame relay and the comparison with X.25.

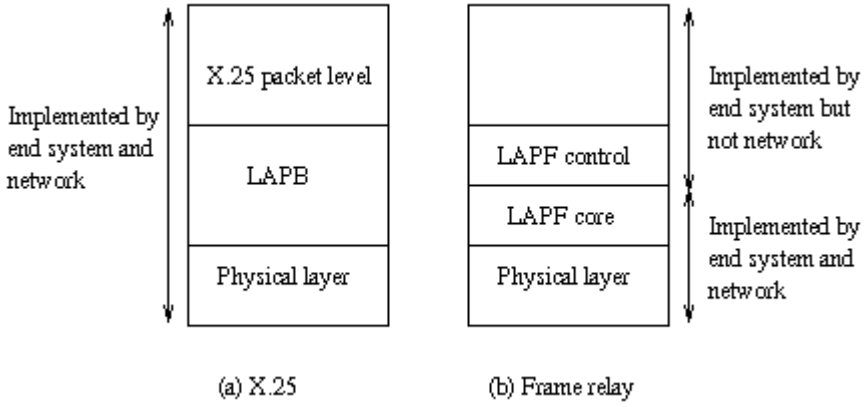


Figure 2.2 Comparison of frame relay and X.25 protocols.

Frame relay involves the physical layer and a data link control protocol known as LAPF (Link Access Protocol for Frame Mode Bearer Services).

Frame relay involves the use of logical connections, called virtual connections. A separate virtual connection is devoted to call control. The setting up and tearing down of virtual connections is down over this permanent control-oriented virtual connection. Other virtual connections are used to carry data only.

Frame relay significantly reduces the amount of work required of the network. However, error recovery is left to the higher layers since a frame in error is simply discarded.

2.4. Congestion in Data Network and Internets

How do congestions occur?

- Congestion occurs when the packets being transmitted through a network begins to approach packet-handling capacity of the network.
- For each node (data network switch and router), there is a queue for each outgoing channel.

- As a rule of thumb, when a channel for which packets are queuing becomes more than 80% utilised, it is an alarming rate of threshold.

2.4.1. Effect of Congestion

Input and output at a given node:

- Any given node has a number of I/O ports attached.
- There are 2 buffers at each port, one for arriving one to hold packets waiting to depart.
- Buffers can be of fixed-size or pool of memories available for all buffering activities.
- As a packet arrives, a node examines each incoming packet, and makes routing decision. Then moves the packet to appropriate output buffer.

Effects of congestion:

- If packets arrive faster than that they can be cleared from outgoing buffers, no memory is available.
- Two general strategies: (1) discard those incoming packets for which there is no available buffer space; (2) apply a sort of flow control over its neighbors so that the traffic flow remains manageable.
- But the neighbors also manage their queues, the congestion points can propagate throughout a region or all of network.

We need a flow control in such a way as to manage traffic on the entire network.

2.4.2. Congestion Control

A number of congestion control techniques exist.

Backpressure

- Similar to backpressure in fluids flowing down a pipe.
- The flow restriction at the end propagates backward (against flow of data traffic) to sources, which are restricted in flow of new packets into network.
- It is limited to connection-oriented network that allows hop-to-hop flow control (such as X.25-based packet-switching networks provide this), neither ATM nor frame relay has the capability.
- IP-based, there is no such facility for regulating the flow of data.

Choke Packet

- It is a control packet generated at a congested node.
- It is transmitted back to a source node to restrict traffic.
- An example is the Internet Control Message Protocol (ICMP) Source Queunch (Message) packet (SQM).

Implicit Congestion Signaling (ICS)

- Two phenomena in network congestion: (1) transmission delayed, and (2) packets are discarded.
- Congestion control on the basis of implicit signaling is the responsibility of end systems that requires all sources be able to detect congestion and reduce flow.
- ICS is efficient in connectionless, or datagram, configuration, IP-based internet.

Explicit Congestion Signaling

- It typically operates over connection-oriented networks and controls the flow of packets over individual connections.
- Two directions:
  1. Backward: -- send alert bit or packets to source.
  2. Forward: -- send alert bit or packets to user then (to source).
- Three categories
  1. Binary: A bit is set in a data packet and is forwarded by congestion node.
  2. Credit based: These schemes are based on providing an explicit credit.
  3. Rate based: To control congestion, any node along the path of the connection can reduce data-rate

2.4.3. Traffic Management

The following issues are related to the efficient use of a network at high load by congestion control mechanisms:

- Fairness – As congestion develops, flows between sources and destinations will experience increasing delay.
- Quality of Service (QoS) – packets may have the different requirements of delay, loss sensitive, throughput etc. A node might transmit higher-priority packets ahead of low-priority packets in the same queue.
- Reservations—is one way to avoid congestion (to provide assured service). Network agrees to give a defined QoS as long as traffic flow is within contract parameters. If net resources are inadequate to meet new reservation.

2.5. Communication Protocol Architectures

2.5.1. The OSI Protocol Architecture

The OSI (Open System Interconnection) Reference Model was developed by ISO (International Standards Organisation) as a model for implementing data communication between cooperating systems. It has seven layers.

- Application: providing end-user applications.

- Presentation: translating the information to be exchanged into terms that are understood by the end systems.
- Session: for the establishment, maintenance, and termination of connections.
- Transport: for reliable transfer of data between end systems.
- Network: for routing the data to its destination and for network addressing.
- Data link: preparing data in frames for transfer over the link and error detection and correction in frames.
- Physical: defining the electrical and mechanical standards and signaling requirements for establishing, maintaining, and terminating connections.

Figure 2.3 depicts the OSI process and peer-to-peer communication, where SDU represent service data unit, Hi (i = 2 … 7) headers of layer i, and T2 is the trailer for layer 2.

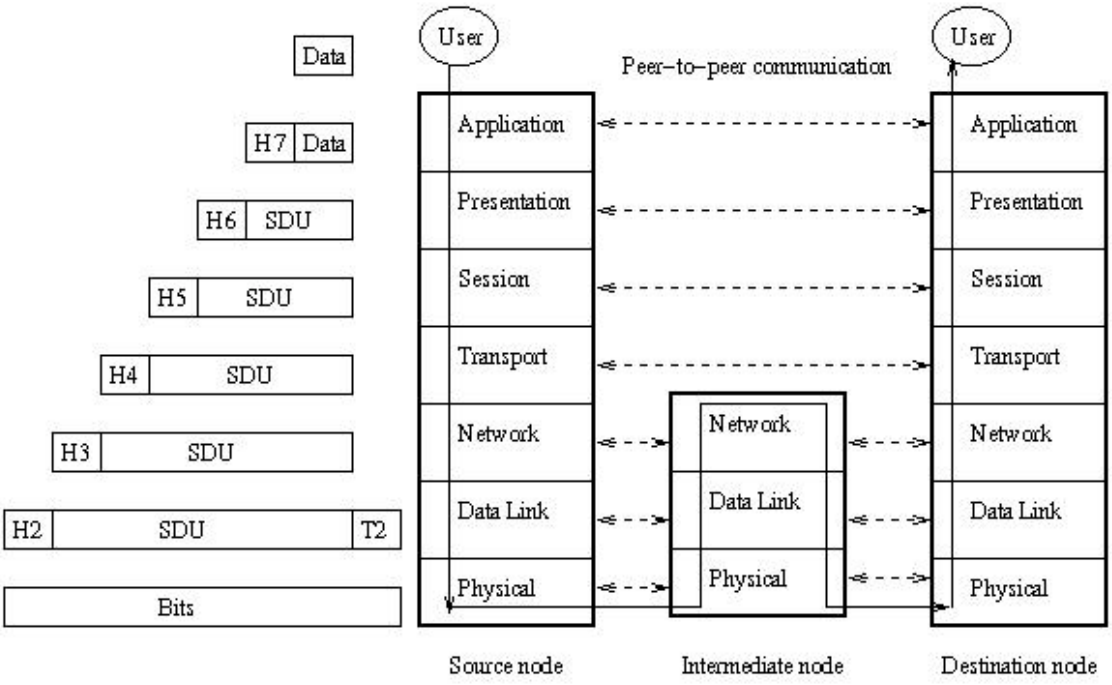


Figure 2.3 OSI process and peer-to-peer communication

2.5.2. Internet Architecture

Internet is the largest data network in the world. It is an interconnection of several packet-switched networks and has a layered architecture. Figure 2.4 shows the comparison of Internet and OSI architectures.

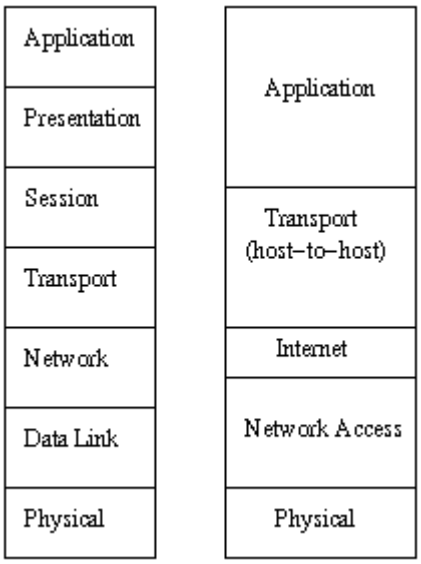


Figure 2.4 Comparison of Internet and OSI architectures

Internet layers:

- Network access layer: It relies on the data link and physical layer protocols of the appropriate network and no specific protocols are defined.
- Internet layer: The Internet Protocol (IP) defined for this layer is a simple connectionless datagram protocol. It offers no error recovery and any error packets are simply discarded.
- Transport layer: Two protocols are defined: the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP is a connection-oriented protocol that permits the reliable transfer of data between the source and destination end users. UDP is a connectionless protocol that offers neither error recovery and nor flow control.
- User process layer (application): It describes the applications and technologies that are used to provide end-user services.

2.5.3. Bridges and Routers

Bridges and routers are intermediate systems that provide a communication path and perform the necessary relaying and routing functions so that data can be exchanged between devices attached to different subnetworks in the internet.

- Bridge: operates at layer 2 of the OSI architecture and acts as a relay of frames between like networks.
- Router: operates at layer 3 of the OSI architecture and route packets between potentially different networks.

Both bridge and router assume the same upper-layer protocols are in use.

2.6. Review Questions

1. Consider a packet-switching network of N nodes, connected by the following topologies:

   - Star: One central node has no attached station; all other nodes attach to the central node.
   - Loop: Each node connects to two other nodes to form a closed loop.
   - Fully connected: Each node is directly connected to all other nodes.

   For each case, give the average number of hops between stations.

2. There is no error detection mechanism in (frame check sequence) in X.25. Isn't this needed to assure that all of the packets are delivered properly?

3. List the major advantages and disadvantages with the layered approach to protocols.

4. Two blue armies are each poised on opposite hills preparing to attack a single red army in the valley. The red army can defeat either of the blue armies separately but will fail to defeat both blue armies if the attack simultaneously. The blue armies communicate through an unreliable communication system (a foot soldier). The commander with one of the blue armies would like to attack at noon. His problem is this: If he sends a messenger to the other blue army, ordering the attack, he cannot be sure the messenger will get through. He could ask for acknowledgment, but that might not get through. Is there a protocol that the two blue armies can use to avoid defeat?

3. Fast Ethernet: IEEE 802.3 100BASE-T

3.1. Study Points

- Understand the IEEE 802.3 100BASE-T protocol family
- Be familiar with the configuration of the 100BASE-T standard
- Understand the strategy of the Gigabit Ethernet

Reference: [STA98] Section 5.1

3.2. Classical Ethernet

3.2.1. CSMA/CD

Classical Ethernet operates at 10Mbps over a bus topology LAN using the CSMA/CD medium access control protocol.

- Bus topology LAN: All stations attach directly to a linear transmission medium (bus). A transmission from any station propagates the length of the medium and can be received by all other stations. Each station has a unique address. Stations transmit data in frames where the destination address is included in the frame header.
- CSMA/CD: A station wishes to transmit first listens to the medium (carrier sense). If the medium is idle, the station may transmit. Collisions are detected and a procedure is used to deal with the situation if a collision occurs.

A more detailed CSMA/CD (carrier sense multiple access with collision detection) protocol is described as follows:

1. Station(s) listen before sending (carrier sense).
2. If the medium is quiet, station(s) start sending (multiple access).
3. If the medium is busy, stations(s) continue to listen until the channel is quiet, then transmit immediately.
4. Transmitting station can detect when the message collides with another station (collision detection).
5. Once a collision is detected, the station stops transmitting and sends a brief jamming signal to inform all stations.
6. After transmitting the jamming signal, the station waits a random amount of time and repeats step 1. The waiting follows the binary exponential backoff algorithm.

The binary exponential backoff algorithm works as follows: A station initiates the retransmission. It also recognises the mean waiting time. If a second attempt of retransmission is needed, the station doubles this mean waiting time and doubles it again on the third attempt, and so on. The aim of this algorithm is to minimise the probability of collision. After a number of unsuccessful attempts, the station gives up and reports an error.

The CSMA/CD protocol works well for light and bursty traffic. For the protocol to be effective, the message frame must be long enough so that the data time is very much longer than the propagation time.

3.2.2. Logical Link Control (LLC)

IEEE defined the LLC in its IEEE802.2 standard. The LLC protocol hides the differences between the various kinds of 802 networks by providing a single format and interface to the network layer. This format, interface, and protocol are all closely based on OSI. LLC forms the upper half of the data link layer, with the MAC sublayer below it. Figure 3.1 depicts the architecture.

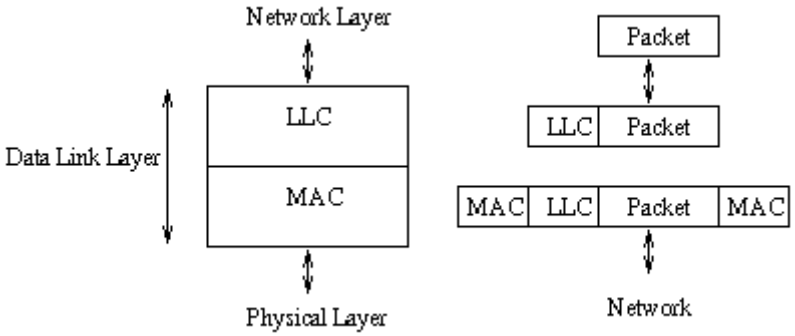


Figure 3.1 LLC and MAC protocols

Typical usage of LLC is as follows. The network layer on the sending machine passes a packet to LLC using the LLC access primitives. The LLC sublayer then adds an LLC header, containing sequence and acknowledgment numbers. the resulting structure is then inserted into the payload field of an 802.x frame and transmitted. At the receiver, the reverse process takes place.

LLC provides three service options: unreliable datagram service, acknowledged datagram service, and reliable connection-oriented service. The LLC header is based on the older HDLC protocol.

3.3. Fast Ethernet

3.3.1. Fast Ethernet Standards

Fast Ethernet (100BASE-T) refers to a set of IEEE 802.3 standards that provides a low-cost, Ethernet-compatible LAN operating at 100 Mbps. Figure 3.2 shows the terminology used.

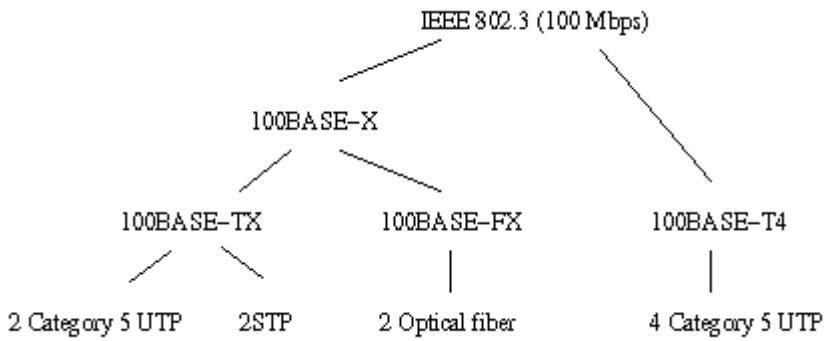


Figure 3.2 IEEE 802.3 100BASE-T options

100BASE-T is the high-speed extension of 10BASE-T. The next table compares the two technologies.

Features	**10BASE-T**	**100BASE-T**
Data rate	10 Mbps	100 Mbps
Standard	IEEE 802.3i	IEEE 802.3u
Topology	Star wired	Star wired
Access method	CSMA/CD	CSMA/CD
Frame size/format	1500 bytes/802.3	1500 bytes/802.3
Encoding	Manchester	4B/5B
Cabling required	UTP Cat. 3/4/5	UTP Cat. 5/STP type
No. of pairs	2	2
Signaling frequency	20 MHz	125 MHz
Distance	100m	100m

Where UTP represents unshielded twisted pairs, STP represents shielded twisted pairs.

Features of various types of 100BASE-T are compared in the next table:

Features	**100BASE-TX**	**100BASE-FX**	**100BASE-T4**
Transmission medium	2 pairs Cat. 5 UTP / 2 pairs STP	2 optical fiber	4 pairs Cat. 3/4/5 UTP
Signaling technique	4B5B, NRZ1	4B5B, NRZ1	8B6T, NRZ
Physical topology	Star	Star	Star
Data rate	100 Mbps	100 Mbps	100 Mbps
Max. segment length	100m	100m	100m
Network span	200m	400m	200m

3.3.2. Cable Standard

EIA/TIA (Electrical Industry Association/Telecommunication Industry Association) is a US standard body that defined cables into five categories:

- Category 1: Typically untwisted 22 AWG or 24 AWG wire with a wide range of impedance and attenuation values. Not for signaling speeds over 1 Mbps.
- Category 2: Similar to Category 1. It uses 22 or 24 AWG solid wire in twisted pairs.
- Category 3: It uses 24 AWG solid wire in twisted pairs and is useful for data transmission at speeds up to 16 Mbps.
- Category 4: It can use 22 or 24 AWG solid wire in twisted pairs. Category 4 has been superseded by Category 5 in most new installations.
- Category 5: This is 22 or 24 AWG unshielded twisted-pair cable and can handle data signaling under specific conditions at 100 Mbps.

3.3.3. Signaling Techniques

- Manchester encoding: A 1 data bit is sent as an uninverted clock pulse, and 0 data bit is sent as an inverted clock pulse.

  The Manchester encoding scheme is self-clocking, because it includes a signal transition in every bit period. But this makes the band rate twice as much as the bit rate of the signal; i.e., the efficiency of the Manchester encoding scheme is only 50%. For a 10 Mbps data rate, a baud rate of 20 Mbps is required.

- 4B5B NRZ1 encoding: The aim of the 4B5B and other encoding schemes used in Fast Ethernet LANs is to improve the efficiency while keeping the scheme self-checking. In the 4B5B scheme, 4-bit code-words are encoded as 5-bit code-words. Of the 32 combinations of the 5-bit code-words, only 16 combinations needed to be used as valid code-words.

  To maintain the self-checking property, only those code words that have at least two transitions in their NRZ1 representation are used. For example, 0000 (a 4-bit code word) is represented by 11110 (a 5-bit code-word).

  16 of the 32 (5-bit) code-words are used to represent data values. Of the remaining code-words, some are used as control codes. The 5-bit code-words that have three or more consecutive 0s are not used.

- 8B6T NRZ encoding: The nomenclature of this code implies that 8-bit binary codes are represented as 6-digit ternary codes.

  In a ternary code, each digit can take three distinct values, can be represented as 0, 1, and 2 (or +, - and 0). A 6-digit ternary code can represent $3^6 = 729$ code combinations. A 8-bit code has 256 distinct code words. This gives a lot of freedom in choosing 256 code words that satisfy the self-checking property.

In the 100BASE-T4 scheme, three lines are used in each direction, each link carrying a 33.3 Mbps data rate. Thus, the baud rate on each link becomes 6/8 * 33.3 = 25 Mbaud.

- 5B6B and 8B10B encoding: These encoding schemes use concepts similar to those described before. 5B6B is usd for the 100VG-AnyLAN specification. 8B10B is used in a 200 Mbaud standard called ESCON, developed by IBM.

3.4. Configuration

3.4.1. Repeaters and Switches

The 100BASE-T network is configured in a star-wire topology, with all stations connected directly to a central point called multiport repeater.

The repeater has the following functions:

- A valid signal appearing on any single input is repeated on all output links.
- If two inputs occur at the same time, a jam signal is transmitted on all links.

Workstations belong to a single CSMA/CD network (such as stations linked to the same repeater) are said in the same collision domain. A bridge is used to link two repeaters to form a larger network. The bridge works in a store-and-forward fashion and therefore participates in two collision domains.

The normal operation of an Ethernet is half duplex in which a station can either send or receive a frame but not both simultaneously. Full duplex operation enables the simultaneous transmit and receive, doubling the data rate of the system.

To support duplex, the stations must have full duplex adapter cards. The Hub must be a switched hub or a bridge. In this case, each station may constitute a separate collision domain. In fact, there is no collision and the CSMA/CD algorithm is no longer needed.

Switching hubs improve Ethernet performance by overcoming the basic limitations of the repeater architecture (i.e., if one is talking, everyone listens). In a switching system, messages pass only to those stations who need them. Other stations remain free to exchange additional traffic.

3.4.2. Mixed 10/100 Mbps Networks

If an organisation wants to introduce 100 Mbps network gradually, there are two products that can boost the system performance:

- 100 Mbps repeater: It provides full 100 Mbps access to a fixed number of users.
- 10 Mbps switch: It benefits all users without upgrading their existing computer hardware.

Figure 3.3 illustrates such an architecture. The 100 Mbps repeater creates a single 100 Mbps collision domain for all the high-performance workstations. The repeater also provides a 10 Mbps switched port bridging internally to the 100 Mbps service. The 10 Mbps switched port then connects to a 10 Mbps repeater, where a number of normal workstations are connected. In the figure, a thick line represents a link with 100 Mbps bandwidth capability, whereas a thin line represents a 10 Mbps bandwidth capability.

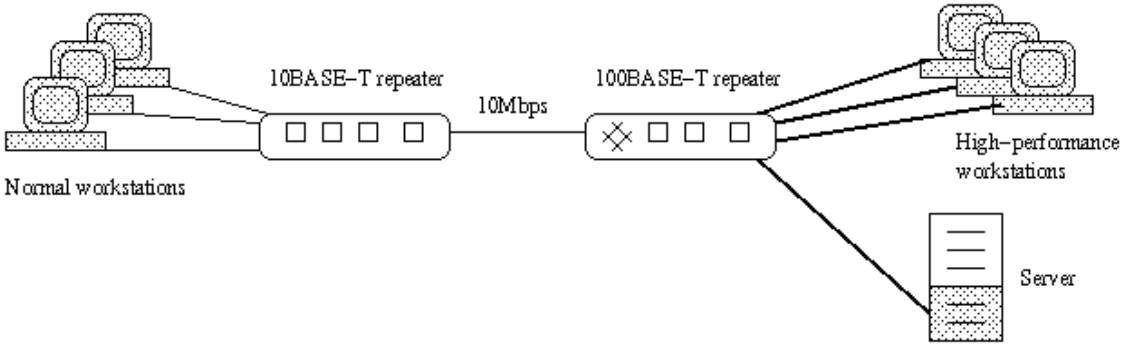


Figure 3.3 A mixed architecture

To extend the 100 Mbps network, a repeater with an expansion port can be used. In Figure 3.4, two repeaters with expansion port (100 Mbps switched) are used to create a two-level hierarchy. Workstations connected to a different repeater belong to different collision domains.

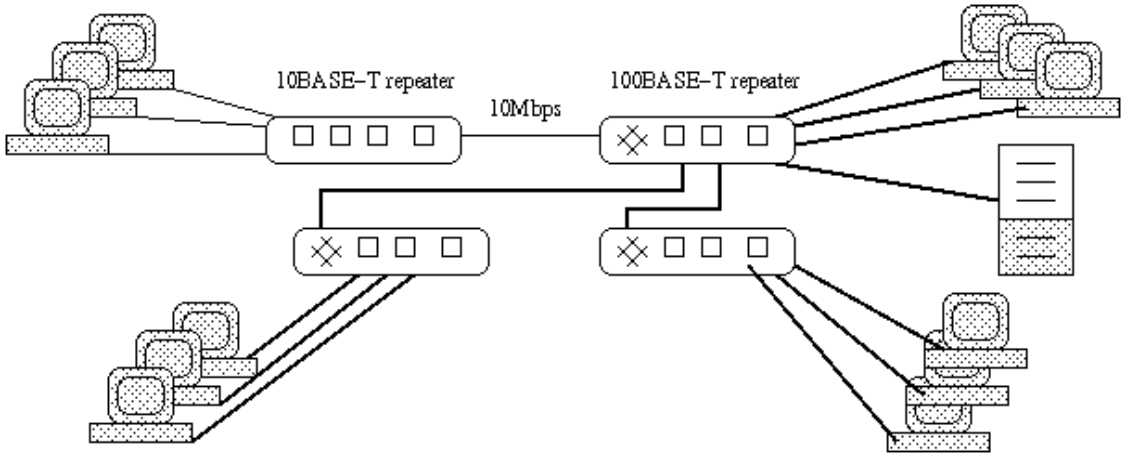


Figure 3.4 Expanding a 100 Mbps network

Another configuration is to use a 10 Mbps switch hub with a 100 Mbps switched port to connect a number of high-performance workstations and a powerful server. The server feeds 100 Mbps into the 100 BASE-T switch port and then the hub fans out to the workstations at 10 Mbps each. Another 10 Mbps repeater can be used to connect other normal workstations to the network. All workstations can benefit the speed boost of the server. Figure 3.5 shows such an example.

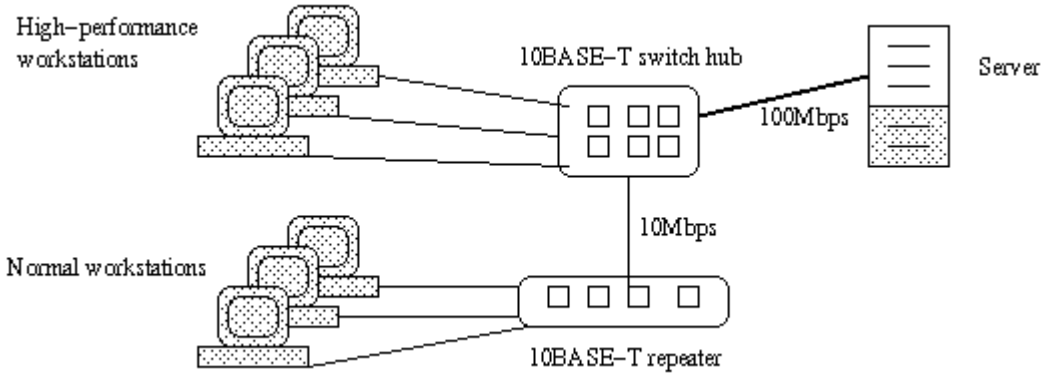


Figure 3.5 Use of a 10 Mbps switch

3.4.3. Pure 100 Mbps Networks

The minimum 100 Mbps network consists of a number of workstations connected to a 100 Mbps repeater (Figure 3.6(a)). As the network grows, the first repeater will eventually run out of ports. At this moment, three basic choices can be followed:

- Add a second repeater: As shown in Figure 3.6(b), workstations in this architecture belong to the same collision domain. Also, not all repeaters can be used in such a two-repeater configuration.
- Use a larger, stackable repeater: A stackable repeater is an expandable repeater which can server a few hundred workstations. Figure 3.6© shows the architecture.

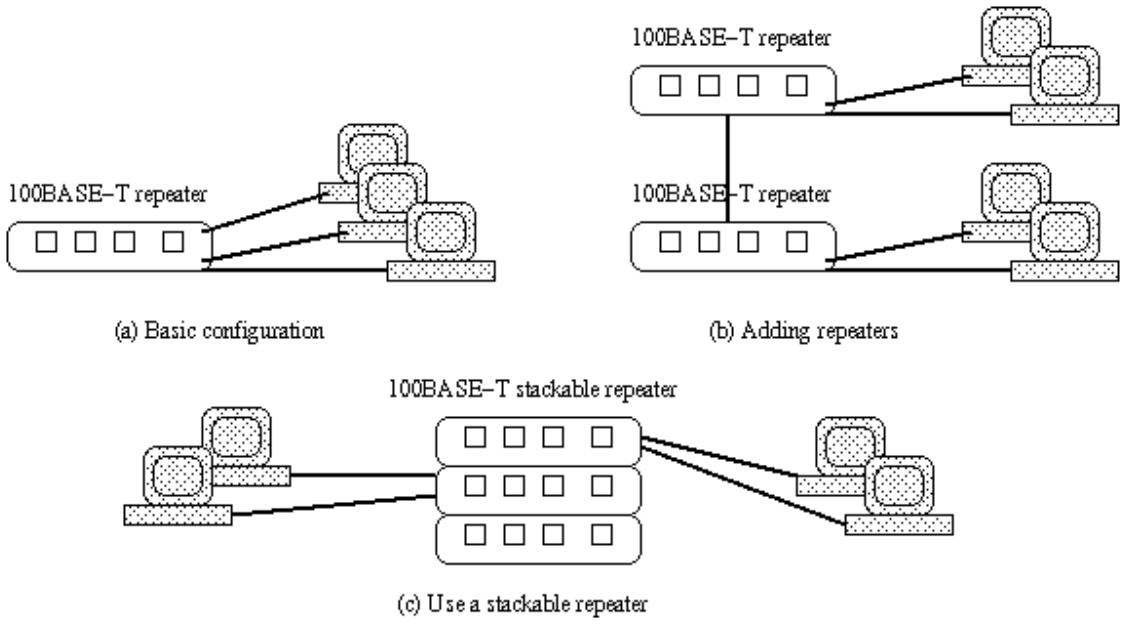


Figure 3.6 Pure Fast Ethernet architectures

- Combine several repeaters with hierarchical expansion ports. To construct very large networks, Fast Ethernet uses switching. In a typical architecture, several repeaters connect to a switch, which in turn connects to a pool of network data servers. Figure 3.7 shows such an architecture.

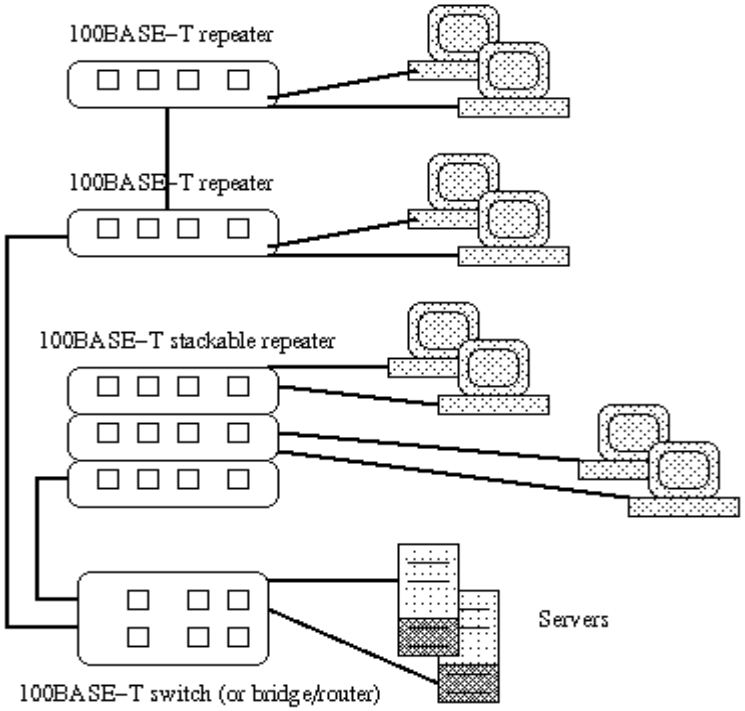


Figure 3.7 A large pure Fast Ethernet architecture

3.5. Gigabit Ethernet

There has been considerate work on the development of the Gigabit Ethernet concept since 1995 when IEEE 802.3 committee formed the high-speed study group. Gigabit Ethernet retains the CSMA/CD protocol and Ethernet format of its 10 Mbps and 100 Mbps predecessors. It is compatible with 100BASE-T and 10BASE-T. The following table compares Gigabit Ethernet with Ethernet and fast Ethernet.

	10BASE-T	**100BASE-T**	**Gigabit Ethernet**
Data rate	10 Mbps	100 Mbps	1 Gbps
Category 5 UTP	100 m	100 m	25-100 m
STP/Coaxial Cable	500 m	100 m	25-100 m
Multimode fiber	2 km	400 m (half duplex) 2 km (full duplex)	500 m
Single-Mode fiber	25 km	20 km	2 km

3.6. Analysis

Fast Ethernet is the natural upgrade path for existing Ethernet users. When configured as a shared-medium network, it provides a ten-fold increase in performance for less than twice the price. In switching configuration, its performance skyrockets another order of magnitude.

3.6.1. Strong Points

- Openness: Many vendors are working together to guarantee interoperation among Fast Ethernet products. Compatible products are widely available.
- High-performance: It provides significant improvement to existing Ethernet.
- Easy to attach to an ATM backbone.
- Dual-speed 10/100 Mbps hubs preserve investiment in 10 Mbps devices, while providing a clear upgrade path to higher performance.
- Support a wide range of cable types.
- Migration strategies for Fast Ethernet are particularly well thought out.
- Minimum investment in network administration since most administrators are already familiar with 10 Mbps Ethernets.

3.6.2. Weak Points

- Ethernet is a non-deterministic system. There are no deterministic guarantees of performance for individual packets. However, switches and bridges can compensate some of the weakness.
- Limited span for the collision domain diameter (fiber: 412 m, UTP cabling: 205 m).

3.7. Review Questions

1. A disadvantage of the contention approach for LANs is the capacity wasted due to stations attempting to access the channel at the same time. Suppose that time is divided into discrete slots with each of N stations attempting to transmit with probability p during each slot. What faction of slots are wasted due to multiple simultaneous transmission attempts.
2. Explain the CSMD/CD protocol with the help of a flow chart.
3. The three aspects of LAN system are topology, medium, and access control. Present the various options for each of these aspects in the form of a classification tree.

4. IEEE 802.8 FDDI and IEEE 802.12 VG-AnyLAN

4.1. Study Points

- Understand the basic architecture of FDDI.
- Understand the traffic types supported by FDDI.
- Be familiar with the basic FDDI protocols.
- Understand the architecture of 100VG-AnyLAN.
- Understand the Demand-Priority algorithm used for 100VG-AnyLAN.
- Be able to compare the differences and similarities of 100VG-AnyLAN and 100BASE-T technologies.

References:

1. "FDDI – A High Speed Network" A. Shah and G. Ramakrishnan, Prentice-Hall, NJ, 1994. ISBN: 0-13-308388-8.
2. Section 3.4 High-Speed Ethernet from "Ethernet Networks," 2nd Ed., G. Held, John Wiley & Sons, Inc., 1996, ISBN: 0-471-12706-X.

4.2. FDDI Concepts

4.2.1. What is FDDI

FDDI stands for Fiber Distributed Data Interface. It is a 100 Mbps data rate LAN standard. The topology is a point-to-point connection of links connected in a logical ring. There are two such rings with one ring configured as a backup. Each station connects to both rings. There can be 500 such stations on the network with a maximum ring-size of 200 km. The maximum distance between each node is 2km.

According to the OSI Reference Model, FDDI specifies layer 1 (physical layer) and part of layer 2 (data link layer). Layer 1 contains FDDI PMD (Physical medium dependent) and PHY (Medium access control) and station management. Layer 2 consists of FDDI MAC (Medium access control) and LLC (Logical link control) sub-layer.

At the current state of standardisation, only one ring is used in normal operation and the second permits faults to be tolerated.

4.2.2. FDDI Nodes and Networking Topology

FDDI specifies a dual ring of tree architecture. There are two such counter-rotating rings (Figure 4.1). The dual ring architecture refers to a *logical* dual ring. Typically, only one ring is used for data transfer (called primary ring). The second ring is used as a backup for fault tolerance.

For example, in Figure 4.1, normal data transfer follows the route of S1 → S2 → S3 → S4 → S1 (the outer ring). However, if the outer ring breaks down, then the inner ring can be used. The data transfer route becomes S1 → S5 → S4 → S3 → S2 → S1. If both ring break down at the same point, say Station S5, due to fire or other accident, then the two rings can be joined at points A and B, respectively, forming a single ring of approximately twice as long.

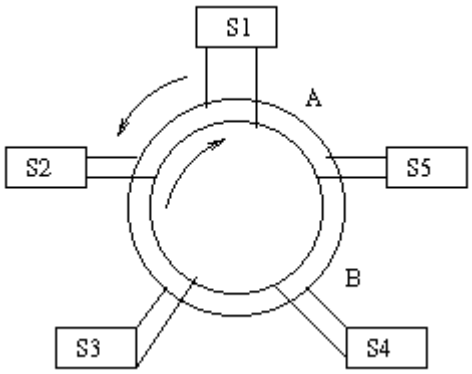


Figure 4.1 FDDI architecture

Definitions:

- *Trunk* ring implies the set of dual rings.
- A *node* is an active element in an FDDI network and it contains at least one port: PHY+PMD.
- A *station* is defined as a node with MAC and it is an addressable entity.
- Dual Attachment Station (DAS) attaches to the trunk ring. A single attachment station (SAS) attaches to one ring only.
- An *optical bypass* or *bypass switch* is a device which provides ring continuity in the presence of a node fault.
- A *concentrator* is a type of a node, which provides connections to single attachment nodes.

Network connections: FDDI defines two classes of stations, A and B. Class A stations connect to both rings. Class B stations connects to only one of the rings.

4.3. FDDI Medium Access Control

4.3.1. Station Types

There are four station types (Figure 4.2):

- Dual attachment station (DAS): The station attaches to the dual ring.
- Dual attachment concentrator (DAC): The concentrator attaches to the dual ring.
- Single attachment station (SAS): The station attaches to the DAC by means of a single ring.

- Single attachment concentrator (SAC): The concentrator attaches to a DAC or another SAC and may support one or more SASs in a single ring.

In FDDI, the physical layer is split into two sublayers:

- Physical medium dependent sublayer (PMD): It provides the optical specification for FDDI such as wavelength, cables, concentrators, power budgets, optical bypass and other details.
- Physical sublayer (PHY): It defines the clock synchronisation, the encoding scheme and other details.

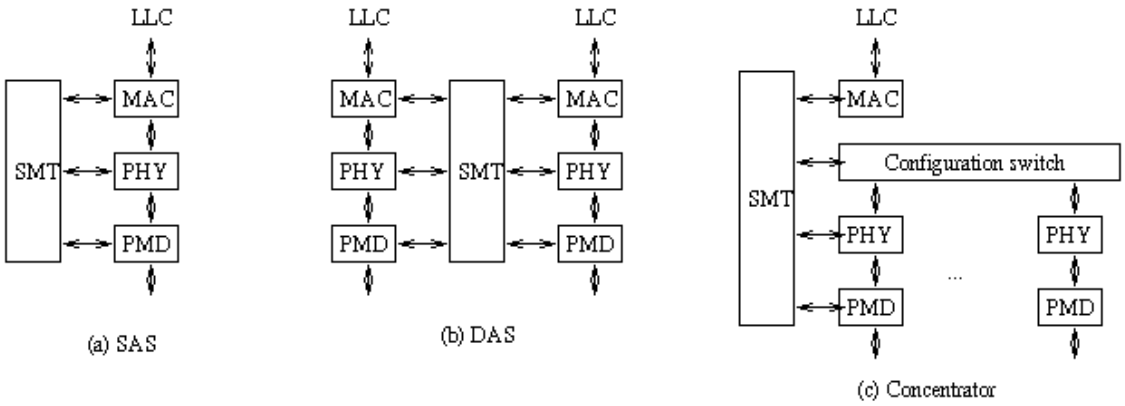


Figure 4.2 FDDI station architctures

4.3.2. FDDI Station Management (SMT)

SMT defines three functions:

- Connection management (CMT): It handles the insertion of stations onto the ring and removal of stations from the ring. This involves establishing or terminating a physical link between adjacent ports and the connection of ports to MAC entities. It consists of:
  - Entity coordination management coordinates the activities of all ports and the optional optical bypass switch associated with that station.
  - Physical connection management handles the point-to-point physical links between adjacent PHY/PMD pairs.
  - Configuration management provides for configuring PHY and MAC entities within a node.
- Ring management (RMT): It receives status information from MAC and connection management, and provides the following services:
  - Stuck beacon detection: A station that suspects a ring failure will transmit a continuous stream of beacons. Eventually, it should receive either a beacon from an upstream station or its own beacon. If neither event occurs, the station will

continue to transmit its own beacon indefinitely (stuck beacon). This is resolved by a stuck-beacon timer which measures the duration of beacon transmission. A stuck beacon procedure is invoked if the timer expires. It transmits a directed beacon to a group of stations forming the ring of the stuck condition.

  - Resolution of problems through the trace process. After the directed beacons are sent and the stuck condition persists, a trace-function is initiated.
  - Detection of duplicated addresses. If two or more MAC entities have the same address, the ring cannot function well. During ring initialisation, the ring is monitored for conditions that indicate the presence of suplicated addresses.
- SMT frame services: It handles the management of stations when the ring is in an operational state. These services are implemented by a set of SMT frames:
  - Neighbourhood information frame.
  - Station information frame.
  - Echo frame.
  - Resource allocation frame.
  - Request denied frame.
  - Status report frame.
  - Parameter management frame.
  - Extended service frame.

4.3.3. FDDI MAC Functions

The MAC layer interacts with the LLC, PHY, and SMT layers and performs important functions such as framing, addressing, error checking, token generation, transmission time control, and fault isolation. The layers LLC, PHY and SMT are pictured as shown in Figure 4.2.

The MAC typically provides at least the following functions:

- Ordered and controlled access to the media by the nodes on the network.
- Address recognition.
- Fair access to the media: No node should utilise an unreasonable proportion of the bandwidth at the expense of other nodes.
- A bound on the network access delay for a node.
- Error checking mechanism.

All LANs provide above functions (such as 802.3 CSMA/CD). FDDI MAC provides the following additional functions and features:

- Multiple classes of traffic
- Dual mode of operation

- Low-level acknowledgment scheme
- Multiple address recognition scheme
- Error detection mechanisms such as frame check sequence (FCS) and error indicators
- Frame handling
- Interface to station management and other upper-layer protocols.

The LLC sublayer is similar for most LANs.

Network Access Mechanism: The protocol adopted for the network access mechanism is the Timed Token Rotation (TTR) protocol. It was felt that ring architecture would be more amenable to providing a low access delay, high throughput, and fault-tolerant channel. There are 3 classes of services in the TTR:

1. Class 1: Guaranteed Bandwidth, *Synchronous* – guarantee of bandwidth and/or deterministic delay and *jitter* characteristics – a variable delay introduced by the network in transmitting each packet. If the first packet takes x ms after being queued and the second packet takes y ms after being queued then the jitter is x-y ms.
2. Class 2: Shared Bandwidth, *Interactive* – for traditional data communications where a minimum throughput is provided but absolute guarantee of bandwidth is not.
3. Class 3: No Guarantee of Bandwidth, *Batch*—left-over bandwidth at light traffic (something of the best effort).

The allocation of bandwidth across the three classes of services was based on the desired token rotation time.

To transmit data, a station has to capture the token. Then it transmits a frame and removes it when it comes around again. A station is allowed to put a token back onto the ring as soon as it has finished transmitting its frames. In a large ring, several frames might be on the ring at the same time.

FDDI MAC Protocol uses three timers. The *token holding timer* determines how long a station may continue to transmit once it has acquired the token. This timer prevents a station from holding the token forever. The *token rotation timer* is restarted every time the token is seen. If this timer expires, it means that the token has not been sighted for too long. Probably it has been lost, so he token recovery procedure is initialised. Finally, the *valid transmission timer* is used to time out and recover from certain transient ring errors.

FDDI also has a priority algorithm that determines which priority class may transmit on a given token pass. If the token is ahead of schedule, all priorities may transmit, but if it is behind schedule, only the highest ones may send.

4.3.4. FDDI Traffic and Capacity Allocation Scheme

FDDI defines two type of traffic: synchronous and asynchronous, and uses the capacity allocation scheme to support the two types of traffic.

Each station is allocated a portion of the total capacity: the frames that it transmits during this time are referred as synchronous frames. Any capacity that is not allocated or that is

allocated but not used is available for the transmission of additional frames, referred to as asynchronous frames.

The capacity allocation scheme defines a target rotation time (TTRT) and each station stores the same value for TTRT. Some or all stations may be provided a synchronous allocation (SAi), which may vary among stations. The allocation must be such that

Dmax + Fmax + Token Time + SAi <= TTRT

where

ASi = synchronous allocation for station i.

Dmax = propagation time for one complete circuit of the ring.

Fmax = time required to transmit a maximum-length frame (4500 bytes).

Token Time = time required to transmit a token.

Initially, each station has a zero allocation and it must request a change in the allocation. Support for SAi is optional. A station that does not support synchronous allocation may only transmit asynchronous traffic.

All station have the same value of TTRT and a separately assigned value of SAi. In addition, each station maintains the following variables:

- Token-rotation timer (TRT)
- Token-holding timer (THT)
- Late counter (LC)

Initially, TRT = TTRT, LC = 0. When the timer is enabled, TRT begins to count down. If a token is received before TRT expired, TRT is reset to TTRT. If TRT counts down to 0 before a token is received, LC is incremented to 1 and TRT is reset to TTRT and again begins to count down. If TRT expires a second time before receiving a token, LC incremented 2, the token is considered lost, and a claim process is initiated.

When a station receives the token, its action will depend on whether the token is early or late. If the token is early, the station saves the remaining time from TRT in THT, resets TRT and enables TRT:

THT ← TRT

TRT ← TTRT

enable TRT.

The station can transmit as follows:

1. It may transmit synchronous frames for a time SAi.
2. After transmitting synchronous frames or if there were no synchronous frames to transmit, THT is enabled. The station may begin to transmit asynchronous frames as long as THT > 0. If the station receives a token and the token is late, LC is reset to 0 and TRT continues to run. The station an then transmit synchronous frames for a time SAi. The station may not transmit any asynchronous frames.

The scheme is designed to ensure that the time between successive sightings of a token is on the order of TTRT or less. Of this time, a certain amount is given to synchronous traffic, others given to asynchronous traffic. As traffic fluctuates, the actual token circulation time may exceed TTRT.

There are 8 levels of priority for asynchronous traffic. Each station has a set of 8 threshold values: TP(1), TP(2), … TP(8) where TP(i) = maximum time a token can take to circulate and still permit priority i frames to be transmitted. The 2nd rule of above can be revised for dealing with asynchronous traffic:

2. After transmitting synchronous frames or if there were no synchronous frames to transmit, THT is enabled and begins to run from its set value. The station may transmit asynchronous data of priority i as long as THT > TP(i). The maximum value of any of the TP(i) must be no longer greater than TTRT.

4.3.5. Ring Monitoring

Ring monotoring responsibility is distributed among all stations. Each station monitors the ring for invalid conditions such as inactivity or incorrect activity. Each station can detect the error condition if the token circulation time > 2 * TTRT. The error detection and correction are done by 3 processes:

- Claim token process (use claim from).
- initialisation process.
- Beacon process (use Beacon frame).

A lost token occurs when a station sets LC=2. The station then initiates the claim token process by issuing a continuous stream of claim frames. The information field of the claim frame contains the station's bid for the value of TTRT. Each claiming station inspects the incoming claim frame and either defers or not according to the followinf guidelines:

1. The frame with the lower TTRT has precedence.
2. Given equal values of TTRT, a frame with 48 bit address has precedence over a frame with a 16 bit address.
3. Given equal value of TTRT and equal address length, the frame with the address of larger numerical value has precedence.

The process completes when one station receives its own claim frame, which has made a complete round without being preempted. At this point, the ring is filled with that station's claim frame and all other stations has yielded. All stations store the value of TTRT contained in the latest claim frame and will be used to allocate capacity.

The station that won the claim token process is responsible for initialising the ring by issuing a non-restricted token. On the first circulation of the token, it is not captured. Each station uses the token for transition from an initialisation station to an operational state, and to reset it TRT.

The beacon frame is used to isolate a serious ring failure such as a break in the ring.

4.4. Analysis of FDDI

Even though the desktop strategy of FDDI did not work, FDDI has been very successful as a backbone for 10 Mbps networks.

4.4.1. Strong Points

- FDDI can span great distance. It is therefore an idea backbone for connecting Ethernet repeaters in a campus setting.
- FDDI uses fiber, which is immune to electromagnetic interference.
- FDDI incorporate redundant features (dual ring).
- FDDI runs 10 times faster than 10 Mbps networking.
- FDDI has deterministic delay performance, as is characteristic for token-based networks.

4.4.2. Weak Points

- Highly complex station management (STM) protocols are required to manage and track FDDI tokens.
- FDDI has not yet announced plans for higher-speed capabilities. Fast Ethernet, ATM, and 100VG-AnyLAN will soon support faster than 100 Mbps.

4.5. VG-AnyLAN Architecture

100VG-AnyLAN was originally proposed by AT&T and Hewlett-Packard as a mechanism to support evolving LAN-based multimedia applications. Later it became the IEEE 802.12 standard, replaced the CSMA/CD access protocol by a demand-priority protocol. AS the name implies, VG stands for voice grade; ANYLAN means to support multiple LAN frame types.

100VG-AnyLAN uses a hierarchical star (cascaded) topology. A central hub, known as a level-1 ot root hub, functions as an inverted tree base in establishing a 100VG-AnyLAN network. From this hub other hubs and / or nodes form a star topology, fanning out underneath the root hub. Figure 4.3 shows the architecture.

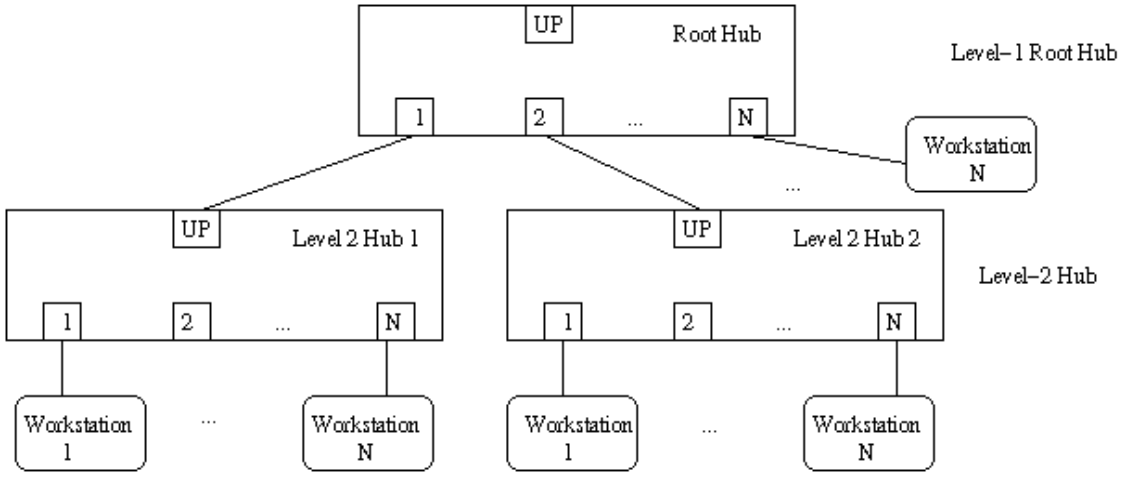


Figure 4.3. 100VG-AnyLAN topology

All hubs located in the same network segment must be configured to support the same frame format—IEEE 802.3 Ethernet or IEEE 802.5 Token-Ring. Through the attachment of a bridge or router to a hub port the 100VG-AnyLAN network can be extended to interconnect with other Ethernet or Token-Ring networks, FDDI- and ATM-based networks, or a wide area network transmission facility.

Each hub in a 100VG-AnyLAN network has one up-link port and N down-link ports. The up-link hub is used to connect lowerlevel hubs to an upper-level hub, while the down-link ports are used to connect an upper-level hub to workstations, bridges, routers, and other network devices to include lower-level hub. Up to three levels of cascading can be used on a 100VG-AnyLAN network.

4.6. Hub Operation

Each hub port can be configured to operate in one of the two modes: normal or monitor. Ports configured in normal mode are forwarded only those packets specifically addressed to the attached nodes. Ports configured in monitor mode are forwarded every packet received by the hub.

Device connected to a hub gain access to a 100VGAnyLAN network through the use of a centrally controlled access method, referred to as demand priority. Under this method, a node isues a request, referred to as a demand, to the hub it is connected to, to transmit a packet onto the network. Each request icludes a priority label assigned by the upper layer application. The priority label is either normal, for normal data packets, or high, for packets carrying time-critical multimedia information. High priority packets are granted access to the network prior to normal priority packets being granted.

The root hub continuously scans its ports using a round-robin sequence for requests. Lower-level hubs connected as nodes also perform a round-robin scanning process and forward node requests to the root hub. The root hub determines which nodes are requesting permission o transmit a packet as well as the priority level associated with the packet. Each hub maintains a separate list for both normal- and high-priority requests.

Normal priority requests are served in their port order until a higher priority request is received. Upon receipt a higher-priority request the hub will complete any packet transmission in progress and then service the high-priority request before returning to the service of the normal-priority list.

To prevent a long sequence of high-priority packets from abnormally delaying low-priority requests from being serviced, the hub also monitors node request-to-end respone times. If the delay exceeds a predefined time, the hub automatically raises the normal-priority level to a high-priority level.

For example, the scanning sequence of Figure 5.1 will be, if all ports have normal priority requests pending:

Level 1 scan: port 1, 2, … N of the Root Hub

Level 2 scan: port 1, 2, … N of the Level-2 Hub 1 and port 1, 2, … N of the Level 2 Hub 2.

When scanning port 1 of the root hub, the sequence of level-2 hub 1 is inserted. The same is true for the sequence of the level-2 hub 2. So the final scanning sequence will be:

port 1, 2, … N of the Level-2 Hub 1, port 1, 2, … N of the Level 2 Hub 2, … port N of the Root Hub.

4.7. Network Layers

4.7.1. DTE Reference Model

Figure 4.4. shows the IEEE802.12 Data Terminal Equipment (DTE) Reference Model.

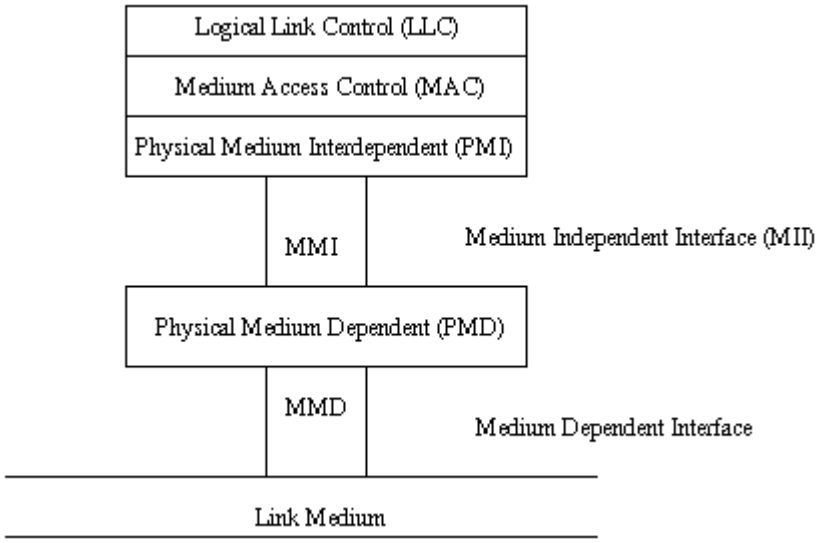


Figure 4.4. IEEE802.12 DTE Reference Model

The IEEE 802.12 subdivided the ISO Data Link Layer into two sublayers: Logical Link Control (LLC) and Medium Access Control (MAC). The LLC sublayer results in the transmission of informaiton in either IEEE 802.3 or 802.5 frame formats, while the MAC

sublayer uses the previously described demand-priority mechanism as the access method to the network. Where the 802.12 differs from the Ethernet and Token-Ring models is in its subdivision of the physical layer into four sublayers.

4.7.2. PMI Sublayer functions

The PMI (Physical Medium Interdependent) sublayer is responsible for performing four key functions prior to passing data to the PMD (Physical Medium Dependent) sublayer. Those functions are:

- Quartet channeling: This is the process of first sequentially dividing MAC frame data octets into 5-bit data quintets. Next, each 5-bit quintet is distributed sequentially among four channels. The rationale for the use of four channels is that they represent a transmission pair for a four UTP demand-priority network.
- Data scrambling: The scramblers used in the PMI layer are employed to reduce the potential affect of radio frequency interference and signal crosstalk between cable pairs.
- 5B6B encoding: The mapping of scrambled 5-bit data quintets into 6-bit symbols is performed by the 5B6B encoder. The encoding process results in the creation of a balanced data pattern that will contain equal numbers of 0s and 1s, providing guaranteed clock transition synchronisation for receiver circuits. In addition, the 5B6B encoding process provides an error-checking capability.
- Preamble data addition: This is to add the preamble and starting and ending frame delimiters to each channel.

The PMD sublayer functions include NRZ encoding, link medium operation and link status control.

4.8. Analysis of 100AnyLAN

4.8.1. Comparison of 100VG-AnyLAN and 100BASE-T

100BASE-T was designed as a mechanism to provide a growth path from 100BASE-T while enabling organisations to retain the invested base of 10BASE-T adapter cards and cabling for workstations that do not require a 100 Mbps operating capability. No provision was made to prioritise traffic. Thus, while a lightly loaded 100BASE-T network can transport multimedia data, this may not be true as the network load increases.

The replacement of the CSMA/CD access protocol by a demand-priority protocol results in 100VG-AnyLAN being more suitable for multimedia applications. Unfortunately, there is a cost associated with the additional technology incorporated into 100VG-AnyLAN, which resulted in network adapters costing more than 100BASE-T adapters.

The next table provides a comparison of the operating characteristics of 100BASE-T and 100VG-AnyLAN.

	100BASE-T	**10VG-AnyLAN**
Data rate	100 Mbps	100 Mbps
Access protocol	CSMA/CD	Demand-priority
Frame support	802.3 Ethernet 802.5 Token-Ring	802.3 Ethernet
Physical topology	Star	Star
Cable support		
2-pair Cat. 5 UTP	100BASE-TX	Not planned
4-pair Cat. 3, 4, or 5 UTP	100BASE-T4	Yes
2-pair STP	100BASE-TX	Future
Fiber	100BASE-FX	Future
Max. UTP drive distance	100m or Cat. 5 UTP	100m Cat. 3, 4, UTP
Max. repeaters allowed	2	4
Max. UTP network diameter	250 m	4,000 m
Full duplex support	Yes	Yes

4.8.2. Strong and Weak Points of 100VG-AnyLAN

Strong points:

- Suitable for providing deterministic performance (e.g. real-time applications) since at heart the 100VG-AnyLAN is a token-passing system.
- 100VG-AnyLAN provides support for 802.3 Ethernet and 802.5 Token Ring packet format (but not both at the same time).
- 100VG-AnyLAN provides a two-level priority scheme, where the video-based applications can use the high priority and others will use the low priority.
- In comparison with a shared bandwidth Fast Ethernet, the 100VG-AnyLAN's maximum usable bandwidth maybe higher, owing to the advantage of its deterministic token-ring like operation.

Weak points:

- The 100VG-AnyLAN suffers from over-complexity. Significant amount of training may be required for network administrators to understand and use its features.
- The medium access layer of 100VG-AnyLAN does not support full-duplex connections. The hub architecture is inherently half duplex.
- Large cascaded networks perform poorly.

- The determinism and priority mechanisms of the 100VG-AnyLAN do not extend through most bridges, switches, or routers.

Now, the majority of network manufactures have adopted Fast Ethernet as the high-speed network of choice. Few companies remain committed exclusively to 100VG-AnyLAN.

4.9. Review Questions

1. Briefly explain the FDDI architecture and its fault tolerant features.
2. What is the capacity allocation in FDDI and how does it work? Use a flow chart to describe the FDDI capacity allocation scheme.
3. Why FDDI uses 4B5B encoding scheme instead of the Manchester encoding scheme?
4. Use an example to illustrate the scanning process of the 100VG-AnyLAN.
5. Briefly explain the architecture of a 100VG-AnyLAN network.
6. What are the major differences of a 100VG-AnyLAN network and a 100BASE-T network?

5. Asynchronous Transfer Mode

5.1. Study Points

- Understand the architecture of ATM: VC and VP.
- ATM Logical Connection
- QoS parameters: delay, loss rate …
- ATM Service Categories
- ATM AAL
- Understand the development of LANs
- Be familiar with the ATM LAN configuration

Understand the interoperability of end systems on various interconnected LANs
References:

1. [STA98] Chapter 4.
2. [STA98] Section 5.2.

5.2. The Architecture

ATM is called cell relay (as compared with frame relay). Frame relay was developed as part of the work of ISDN, but is now finding wide application in private networks and non-ISDN applications in routers.

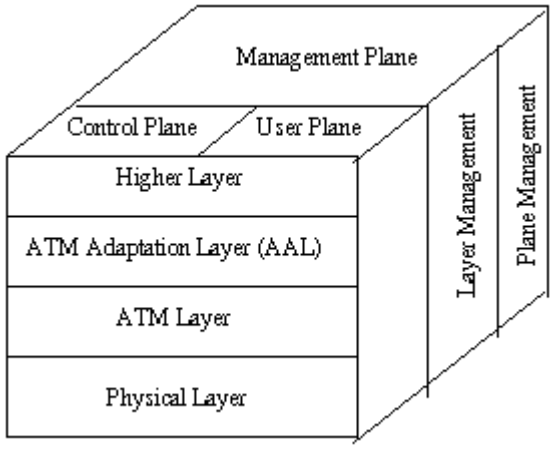


Figure 5.1 ATM protocol architecture

Figure 5.1 depicts the ATM Protocol Architecture. ATM, like packet switching and frame relay, allows multiple logical connections to be multiplexed over a single physical interface.

User plane provides for user information transfer, along with associated control (e.g., flow control, error control).

Control plane performs call control and connection control functions.

Management plane includes plane management, which performs management functions related to a system as a whole and provides coordination between all the planes. Layer management performs management functions relating to resources and parameters residing in its protocol entities.

5.3. ATM Logical Connection

5.3.1. Virtual Channel Connection

Virtual channel (VC) is a generic term used to describe unidirectional transport of ATM cells associated by a common unique identifier value. Logical Connections in ATM are referred to as virtual channel connections (VCCs). A VCC is set up between two end users through the network, and a variable-rate, full-duplex flow of fixed size cells is exchanged over the connection.

The second concept introduced is the Virtual Path Connection (VPC). The VPC has several advantages:

- Simplified network architecture: Network transport functions can be separated into those related to an individual logical connection (VC) and VP.
- Increased network performance and reliability: The network deals with fewer, aggregated entities.
- Reduced processing and short connection setup time: By reserving capacity on a virtual path of later arrivals, new VCC can be established by executing simple control functions at the end points of VPC.
- Enhanced network services: VP is used internal to the network but is also visible to the end user.

The process of setting up a VPC is decoupled from the process of setting up an individual VCC.

VPC control mechanisms include calculating routes, allocating capacity, and sorting connection state information.

To set up VC, there must first be a VP to the required destination node with sufficient available capacity to support the virtual channel, with appropriate QoS.

5.3.2. VCC Use

- Between end users: Can be used to carry end-to-end user data; to carry control signaling between end users. VCC should not exceed VPC capacity.
- Between an end user and a network entity: Used for user-to-network control signaling. Can be used to aggregate traffic from an end user to a network exchange or network server.

- Between two network entities: Used for network traffic management and routing functions. A network-to-network VPC can be used to define a common route for the exchange of net management information.

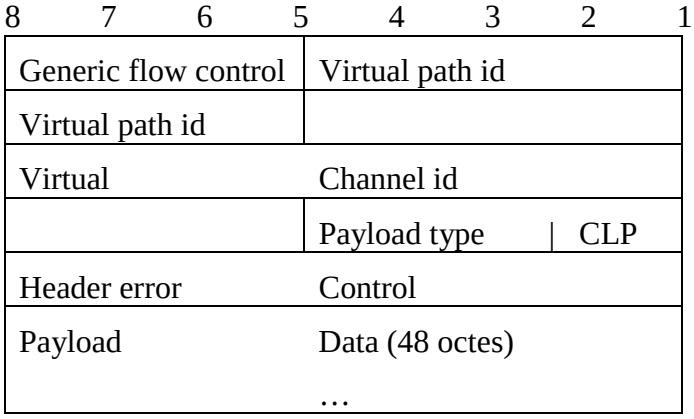
5.3.3. VP/VCC Characteristics:

- Quality of service: parameters such as cell loss ratio and cell delay variation.
- Switched and semi-permanent VCC is an on demand connection, which requires call-control singling for setup and tearing down. A semi-permanent VCC is one that is of long duration and is set up by configuration or network management action.
- Cell sequence integrity: A sequence of transmitted cell within a VCC is preserved.
- Traffic parameter negotiation and usage monitor: Traffic parameter can be negotiated between user and network. The input of cells is monitored by the network to ensure that the negotiated parameters are not violated.
- Virtual channel identifier restriction within a VPC: One or more virtual channel identifiers or numbers may not be available to the user of the VPC, but may be reserved for network use.

5.4. ATM Cells

5.4.1. Header Format:

1. User – network interface



3. Network – network interface

8	7	6	5	4	3	2	1
Virtual			path id				
Virtual path id							
Virtual			Channel id				
			Payload type				CLP
Header error			Control				
Payload			Data (48 octets – information field)…				

Generic flow control—could be used to assist the customer in controlling the flow of traffic for different QoS. One candidate for the use of this field is a multiple-priority level indicator to control the flow of info in a service dependent manner.

Virtual path identifier (VPI) – constitutes a routing field for the network. It is an 8-bits at the user-network interface and 12-bits at the network-network interface.

Virtual channel identifier (VCI) – service access point, is used for routing to and from the end user.

Payload type (PT) – indicates the type of information in payload field.

PT Coding	Interpretation
000	User data cell, congestion not experienced, SDU type =0
001	User data cell, congestion not experienced, SDU type =1
010	User data cell, congestion experienced, SDU type =0
011	User data cell, congestion experienced, SDU type =1
100	OAM segment associated cell
101	OAM end-to-end associated cell
110	Resource management cell
111	Reserved for future function

Where SDU = Service Data Unit. OAM = Operations, Administration, and Maintenance.

Cell loss priority (CLP) is used to provide guidance to the network in the event of congestion. 0 -- high priority, should not be discarded unless no other alternative is available. 1 -- this cell can be discarded in case of congestion.

5.4.2. Header Error Control

Each ATM cell includes an 8-bit header error control (HEC) field that is calculated based on the remaining 32 bits of the header. The HEC uses the polynomial $x^8 + x^2 + x + 1$ to generate the code. As the input to the calculation is only 32 bits, it not only allows the code to be used to detect errors, but also to correct them. Figure 5.2 shows the HEC operations at the receiver.

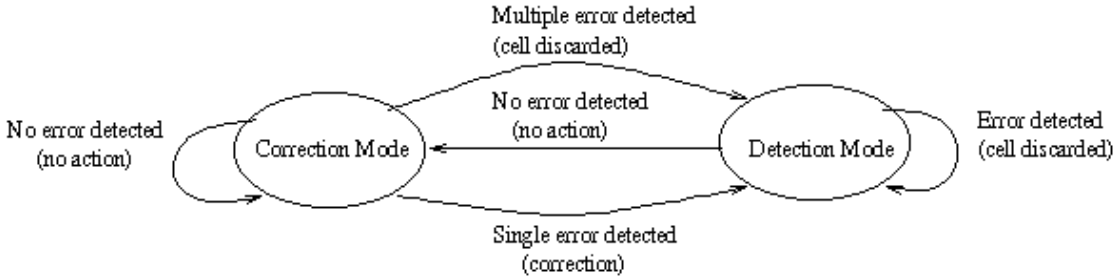


Figure 5.2 HEC operations at receiver

The operations consist of two modes: correction mode and detection mode. Initially he receiver is at correction mode. When a cell arrives, HEC is calculated. If no errors, the receiver remains in correction mode. If an error is detected, and it is a single-bit error, the receiver will correct it and moves to detection mode. If the error is a multibit error, the receiver just moves to the detection mode.

No error correction is made in detection mode. The receiver remains in detection mode as long as errored cells are received. When a good cell is received, the receiver switches back to correction mode.

5.5. ATM Service Categories

ATM network is designed to be able to transfer many different types of traffic simultaneously, including real-time flows such as voice, video, and burst TCP flows.

The following service categories have been defined by the ATM Forum:

- Real-time service
  - Constant bit rate (CBR)
  - Real-time variable bit rate (rt-VBR)
- Non Real-time service
  - Non-real-time variable bit rate (nrt-VBR)
  - Available bit rate (ABR)
  - Unspecified bit rate (UBR)

5.5.1. Real-Time Service

Real-time applications typically involve a flow of information to the user that is intended to reproduce the flow at a source. The demand in the ATM network for switching and delivery of real-time data are high.

The constant bit rate (CBR) is used by applications that require a fixed data rate that is continuously available during the connection lifetime and a relatively tight upper bound on transfer delay. Examples of CBR applications include:

- Videoconferencing
- Interactive audio (telephony)
- Audio/video distribution (television, distance learning, pay-per view)
- Audio/video retrieval (video-on-demand, audio library)

The real-time variable bit rate (rt-VBR) is intended for time sensitive applications, i.e., those require tightly constrained delay and delay variation.

5.5.2. Non Real-Time Service

Non-real-time services are intended for applications that have bursty traffic characteristics and do not have tight constraints on delay and delay variation.

With the non-real-time variable bit rate (nrt-VBR) service, the end system specifies some traffic characteristics. The network can then provide substantial improved QoS. Example applications include airline reservations, banking transactions and process monitoring.

The unspecified bit rate (UBR) service is a best effort service. With UBR, cells are forwarded on a FIFO basis using the capacity not consumed by ither services. Example applications include FTP, information distribution and retrieval, and remote terminal.

With the available bit rate (ABR) service, an application specifies a peak cell rate (PCR) and a minimum cell rate (MCR). The network allocates resources so that all ABR applications receive at least their MCR capacity. Any unused capacity is then shared among all ABR sources.

5.6. ATM Adaptation Layer

The use of ATM creates the need for an adoption layer to support information transfer protocols not based on ATM.

5.6.1. AAL Services

- Handling of transmission errors
- Segmentation and reassembly, to enable larger blocks of data to be carried in the information field of ATM cells
- Handling of lost and misinserted cell conditions
- Flow control and timing control

5.6.2. Service Classification for AAL

	Class A	Class B	Class C	Class D
Timing relation between source and destination	Required		Not Required	
Bit Rate	Constant	Variable		
Connection Mode	Connection Oriented			Connectionless
AAL Protocol	Type 1	Type 2	Type ¾	
			Type 5	

5.6.3. AAL Protocols

AAL layer is organised in two logical sublayers:

- Convergence sublayer (CS) – provides the functions needed to support specific application using AAL.
- Segmentation and reassembly sublayer (SAR) – Each AAL user attaches to AAL at a service access point (SAR).

Initially, ITU defined one protocol type for each class of service (AAL type 1 though Type 4). Each protocol type consists of two protocols, one at CS and one at SAR sublayer. More recently, types 3 and 4 were merged into a Type ¾ , and a new type, Type 5, was defined.

5.7. ATM LANs

5.7.1. Development of LANs

There are 3 generations of LANs:

- First generation: Typified by CSMA/CD and token ring LANs. The first generation provided terminal-to-host connectivity and supported client-server architectures at moderate rates.
- Second generation: 100-Mbps Fiber Distributed Data Interface (FDDI), an optical token ring LAN support high-performance workstations.
- Third generation: is designed to provide the aggregate throughput and real-time transport guarantees that are needed for multimedia applications.

Requirements for the 3rd generation of LANs:

- Support multiple, guaranteed classes of service. For example, a live video application may require a guaranteed 2-Mbps connection for acceptable performance while FTP can use “background” class of service.

- Provide scalable throughput that is capable of growing both per-host capacity and aggregate capacity.
- Facilitate the interworking between LAN and WAN technologies.

ATM is ideally suited to these requirements:

- ATM using VC and VP, multiple classes of service are easily accommodated.
- ATM is easy scalable by adding more ATM switching nodes using higher/lower data rates for attached devices.

Possible types of ATM LANs are the following:

- Gateway to ATM WAN
- Backbone ATM switch
- Workgroup ATM

In practice, a mixture of two or all three of the above types of networks is used to create an ATM LAN.

ATM LAN uses centralised hubs connected to stations with start wiring. The station links may run at 155 Mbps, 51 Mbps, or 25 Mbps, depending on the link technology employed. The same links are used as backbone connections.

5.7.2. ATM LAN Configurations

ATM LAN can be configured using ATM switches (nodes) or ATM hubs. An examples of ATM LAN configuration using switches includes

- ATM switches interconnected with high-speed point-to-point links
- Various types of LANs linked to the ATM switches
- [STA98] page 116, Figure 5.9 presents an example of this configuration.

Functions of ATM switches include:

- communicating with other ATM swithes in ATM cell streams
- buffering and speed conversion between the attached LAN and the ATM data rate
- protocol conversion from the MAC protocol used on the attached LAN to the ATM cell stream used on the ATM network

If an ATM LAN is configured using ATM hubs, then

- Each ATM hub includes a number of ports that operate at different data rates and use different protocols.
- Each end system has a dedicated point-to-point link to the hub.
- An example of this configuration is Figure 5.10, page 117 of [STA98].

An important issue in ATM LAN configuration is the interoperability of interconnected LANs. That includes:

- Interaction between an end system on an ATM network and an end system on a legacy LAN
- Interaction between an end system on a legacy LAN and an end system on another legacy LAN of the same type
- Interaction between an end system on a legacy LAN and an end system on another legacy LAN of a different type

A possible solution: use a bridge plus protocol mapping between various MAC layers and mapping between MAC formats and ATM cells.

5.7.3. Analysis of ATM LANs

ATM at the local level is an attractive option primarily because of its perceived capability to provide wide area ATM connectivity once wide area ATM networks are deployed. During the next few years, before the ATM WAN becomes a reality, users will face a choice between ATM, for its perceived future WAN compatibility, and other 100 Mbps networks, with their greater availability, greater range of options and configurations, and software support.

ATM LAN strong points include:

- ATM LAN should integrate smoothly with future wide area ATM services.
- The small ATM cell size provides very low delay.
- ATM links can operate at a range of speeds spanning upward of several gigabits per second. ATM also uses a switched architecture. These two features provide almost unlimited bandwidth capability for future expansion.
- ATM links operate over a wide range of cables.

ATM LAN weak points include:

- ATM specifications are not yet complete.
- Once ATM standards are completed, their benefits at the local level will have been seriously eroded by advances in 100 Mbps networks.
- ATM LAN also must face some very tough traffic issues. For example, the original ATM traffic models assumed that most devices would generate fine, continuous streams of tiny packets. When aggregated, thousands of these fine stream were suppose to merge into large, smooth rivers of data. The smoothness helps minimise ATM switch buffer requirements, helps reducing the cost of ATM networks. However, the flow of data in a LAN is inherently jerky and there are not enough devices present in a typical LAN to average out the jerky flow of data. As a result, the traffic calculations for typical LAN applications do not favour ATM switch designs.

5.8. Review Questions

1. One method of transmitting ATM cells is as a continuous stream of cells, with no frame imposed; therefore, the transmission is simply a stream of bits, with all bits

being parts of cells. Because there is no external frame, some other form of synchronisation is required. This can be achieved using the HEC function. The requirement is to assure that the receiver knows the beginning and ending cell boundaries and does not drift with respect to the sender. Draw a state diagram for the use of HEC to achieve cell synchronisation, and explain the functionality.

2. Briefly explain the ATM architecture.
3. What are the services of the ATM adaptation layer? Explain how can these services be used to support information protocols not based on ATM.
4. Use an example to illustrate the HEC operation at receiver.
5. Briefly explain various types of ATM LAN configuration.
6. Briefly explain the strong and weak points of ATM LAN.

6. High Speed Transport-Level Traffic Control

6.1. Study Points

- Understand the operation of transmission control protocols
- Understand the congestion control in TCP
- Be familiar wit the performance of TCP over ATM
- Understand the principles of real-time transport protocols

Reference: [STA98] Chapter 10.

6.2. Transmission Control Protocol

Transport protocol is a vital ingredient in the design and implementation of communication networks. It provides an interface between applications and network and enables applications to request QoS.

Transmission control protocol (TCP) is the most widely used transport protocol.

6.2.1. TCP Header Format

The header contains all information for protocol mechanisms. The minimum number of the header is 20 octets.

- Source port: 16 bits, source TCP user.
- Destination port: 16 bits, destination TCP user.
- Sequence number: 32 bits, if SYN flag is set, this field is the initial sequence number (ISN) and the 1^{st} octet is ISN+1.
- Acknowledgment number: 32 bits, a piggybacked acknowledgment. Contains the sequence number of next data octet that the TCP entity expects to receive.
- Data offset: 4 bits.
- Reserved: 6 bits.
- Flags: 6 bits, URG: urgent; ACK: acknowledgment; PSH: push function; RST : reset the connection; SYN: synchronise; FIN: no more data from sender.
- Window: 16 bits, flow control credit allocation in octets.
- Checksum: 16 bits.
- Urgent Pointer: 16 bits.
- Options: variable.

6.2.2. TCP Flow Control

TCP uses a form of sliding window mechanism, known as a credit allocation scheme. A TCP entity acknowledges an incoming segment data field (A=i, W=j) where

- All octets through sequence number i-1 are acknowledged, the next expected octets has sequence number i.
- Permission is granted to send an additional window (W), j octets correspond to sequence numbers i through i+j-1.

The credit allocation scheme is quite flexible. For example, if the last message issued by B was (A=i, W=j); the last octet of data received by B was octet number (i-1), then

- To increase credit to k (no additional data have arrived) B issues (A=i, W=k)
- To acknowledge an incoming segment, containing m octets (*m<j)* without granting additional credit, B issues (A=i+m, W =j-m).

6.2.3. Effect of Window Size on Performance

Let

- W=TCP window size (octets).
- R = Data rate (bps) at TCP source available to a given TCP connection.
- D = Propagation delay (seconds) between TCP source and destination over a given TCP connection.

Suppose a TCP entity begins to transmit a sequence of octets over a connection. It takes *D* seconds for the 1^{st} octet to arrive.

Let a source to transmit a sequence of octets over a connection to a destination. it needs *D* seconds to destination, and additional *D* for acknowledgment to return.

During this 2*D* time, 2RD bits or RD/4 octets can be transmitted. If W>RD/4, then full throughput can be achieved, whereas if W<RD/4, then ratio 4W/RD is achieved.

Examples of effect of window scale parameters:

- For a 1 Gbps Ethernet, then 10^3 bits is maximum with T-1 trunk (1.55 Mbps).
- SDH – (synchronous digital hierarchy) 155 Mbps between 2 end points with window size $2^{20}-1 = 10^6$.

6.2.4. Adaptive Retransmission Timer

Set timer great than round-trip delay: $ARTT(K+1) = 1/(K+1)\Sigma_{I=1\ldots K+1}RTT(i)$ where RTT(i) is round trip time for the ith transmitted segment and ARTT(K) is the average round-trip time for K segments.

ARTT(K+1) = K/(K+1)ARTT(K) +1/(K+1)RTT(K+1) -- It is not necessary to recalculate the entire sum each time.

Give greater weight to more recent instances because it is more likely reflect the future behavior:

Smoothed round-trip time: $SRTT(K+1) = \alpha \times SRTT(K) + (1- \alpha) \times RTT(K+1)$

$SRTT(K+1) = \alpha \times SRTT(K) + (1- \alpha) \times RTT(K+1) \quad 0<\alpha<1.$

Timer should be set somewhat greater than rtt: $RTO(K+1) = SRTT(K+1) + \Delta$

$RTO(K+1) = MIN(UBOUND, MAX(LBOUND, \beta \times SRTT(K+1))$

6.2.5. TCP Implementation Policy Options

- Send Policy: In the absence of PUSHed data and a closed transmission window, a sending TCP entity is free to transmit data at its own convenience.
- Deliver Policy: In the absence of a PUSH, a receiving TCP is free to deliver data to the user at its own convenience.
- Accept Policy: TCP places the data in a receive buffer. Receiving TCP entity has two options:
  - In order: Accept only segments arrive in order. Out-of order will be discarded
  - In window: Accept all segs within receive window.
- Retransmit Policy: 1st only; Batch; Individual.
- Acknowledge Policy: Immediate; Cumulative.

6.3. TCP Congestion Control

6.3.1. Congestion Control in TCP-Based Internet

Congestion reduces availability, throughput, and response time. Dynamic routing can help to alleviate congestion by spreading the load more evenly among the switches and links.

TCP/IP congestion control is a difficult for the following factors:

- IP is a connectionless, stateless protocol, no provision for detecting much less for controlling congestion.
- TCP provides end-end flow control;
- There is no cooperative, distributed algorithm to bind together the various TCP entities. TCP entities cannot cooperate to maintain total flow control but compete selfishly for available resources.

For the connectionless IP environment, the ICMP Source Quench message provides a blunt and crude instrument for restraining source flow. RSVP (resource reservation protocol) may be a way for such systems.

The only tool in TCP that relates to the network congestion control is the sliding-window flow and error control mechanism. A number of cliever techniques have been developed that enable the use of this mechanism for congestion detection, avoidance, and recovery.

6.3.2. Retransmission Time Management

The value of the retransmission timer (RTO) can have a critical effect on TCP's reaction to congestion. The techniques that deal with the calculation of the RTO are:

- RTT (round-trip time) variance estimation (Jacobson's algorithm): to estimate the variability in RTT values and to use that as input into the calculation of an RTO. Experience has shown that Jacobson's algorithm can significantly improve TCP performance.
- Exponential RTO backoff: used to calculate the RTO value used on retransmission segment.
- Karn's algorithm: used to find out which round-trip samples should be used as input to Jacobson's algorithm.

6.3.3. Window Management

The size of the TCP's send window can have a critical effect on whether TCP can be used effectively without causing congestion. Four techniques:

Slow start. TCP transmission is constrained by the following relationship: *awnd = MIN[credit, cwnd]*

where *awnd* = allowed window, in segments. This is the number of segments that TCP is currently allowed to send without receiving further acknowledgment.

cwnd = congestion window, in segments. A window used by TCP during startup and to reduce flow during periods of congestion.

credit = the amount of unused credit granted in the most recent acknowledgment in segments. When an acknowledgment is received, this value is calculated as *window/segment size*.

- Dynamic window sizing on congestion:

  Slow-start has been found to work effectively for initialising a connection. However, once congestion occurs, it may take a long time for the congestion to clear. Thus the exponential growth of congestion window under slow start may be too aggressive and may worsen the congestion. Instead, Jacobson proposed the use of slow start to begin with, followed by a linear growth in congestion window.

- Fast retransmit: the retransmission timer (RTO) is general longer than RTT.

  1. RTO is calculated on basis of predication of next RTT. Could be small.
  2. If delays at destination fluctuate, estimated RTT becomes unreliable.
  3. Destination may cumulatively ACK multiple segments.

- Fast recovery: retransmit the lost segment, cut the congestion window in half, and then proceed with the linear increase of congestion window. This technique avoids the initial exponential slow-start process.

6.4. Performance of TCP over ATM

The essential issue is how best to manage TCP's segment size, window management, and congestion control policies on the one hand, and ATM's QoS and traffic control policies on the other, to achieve high throughput for TCP traffic, fair allocation among various TCP connections, and efficient use of underlying ATM network.

6.4.1. Protocol Architecture

The typical implementation of TCP/IP over ATM uses the following protocol stack:

TCP → IP → AAL5 →ATM

- AAL maps each higher-layer protocol data unit into ATM cells.
- AAL has 2 sublayers: Convergence Sublayer (CS) provides functions to support specific higher layer and Segmentation & Reassembly Sublayer (SAR) is responsible for packing info. from CS into cells and unpacking info at other end.
- AAL5 – most commonly used form of AAL to support TCP/IP traffic.
- AAL5-- no overhead bits at SAR level, simply segs. CS PDU
- CS PDU is segmented into 48 octets.

6.4.2. TCP over UBR

ABR (available bit rate) and UBR (unspecified bit rate) are designed to support traditional data traffic applications, not voice and video. ABR is intended for applications in which delay is a concern. UBR is directed at delay-tolerant applications.

Users concern throughput, network concerns bursts of traffic.

The main practical difference between ABR and UBR is that, in the case of ABR, the network will provide congestion information to the user, enabling the user to reduce or increase the sending rate to achieve high efficiency. No such feedback is provided for UBR. However, because the ABR is relatively new and is considerably more complex than UBR, UBR has been the preferred service for TCP traffic.

Buffer capacity at ATM is a critical parameter in accessing TCP throughput performance. Theoretical and experimental analysis show that:

- Smaller switch buffer size reduces throughput.
- Larger TCP segment size also reduces throughput.
- Larger TCP receive window size reduces throughput.
- Increased congestion, caused by an increased number of TCP connections, reduces throughput.

The key to understanding the difference between cell-based and packet-based switching performance is: dropping a cell causes other cells of the same IP datagram unusable. Yet the ATM will forward these useless cells to the destination. Therefore:

- Smaller buffers increase the probability of dropped cells.
- A larger segment size increases the number of useless cells that are transmitted if a single cell is dropped.

Two strategies to reduce the transmission of useless cells:

- Partial Packet Discard (PPD): if a cell from IP datagram is dropped, then following pkts are all dropped.
- Early Packet Discard (EPD): when a switch buffer reaches threshold, before it actually drops cells, then entire packet is dropped.

Combination of EDP with TCP congestion control mechanism provides effective throughput for average TCP connection.

EPD with Fair Buffer Allocation

EPD is effective, but *fairness* is a serious problem.

EDP is exhibited unfair treatment of TCP connections.

- EDP has a bias against connections with shorter IP datagram – when ATM finds a packet to drop, it is likely it drops the short packet.
- EDP has a bias against connections that pass through multiple congested switches.

To increase fairness: use fair buffer allocation (FBA) with EDP. When EDP is invoked, switch will choose to discard from VC that using more than its fair share of switch's buffer. Let

B: buffer capacity, in cells

R: threshold triggers packet discard R<B

N: number of cells in B

N(i) – number of cells in buffer for VC i

V: number of active VCs, i.e. the number of VCs that have at least one cell in buffer.

So, $N = \Sigma_{I=1..V} N(i)$. Ideally, there would be N/V cells buffered for each VC.

Weight W(i) for each VC: $W(i) = N(i)/(N/V) = N(i)V/N$

W(i) > 1 then VC i has disproportionate share of buffer.

Selective drop: → (N>R) & W(i) > Z (Z >1) – provides improved fairness ←→ EDP → force TCP to reduce the window size, work with TCP control to balance loads.

- Fairness and total throughput increase with increased ATM switch buffer size.
- Fairness decreases with increase of sources.

Overall normalized throughput = $\Sigma x_i / (V*M)$; x_i = throughput of the ith TCP source; where V= number of VCs (TCP connections); M = max. possible TCP throughput.

$$\text{Fairness} = (\Sigma x_i)^2 / (V*\Sigma(x_i^2))$$

6.4.3. TCP Over ABR

Assessment of TCP performance over ABR is even more complex than the TCP-UBR case.

ABR flow control has two modes:

- Binary: switch signals the beginning/end of congestion to the sources.
- Explicit rate mode: an ATM switch employs a control algorithm to allocate capacity among the VCs traversing the switch.

Performance of TCP over ABR—important conclusion:

- It is quite sensitive to some ABR parameter settings.
- Overall ABR does not provide significant performance improvement over the simple UBR-EDP-FBA.

6.5. Real-time Transport Protocol (RTP)

6.5.1. Limitations of TCP in RT Applications

RT applications include audio/video conferencing, live video distribution, shared workspaces, remote medical diagnosis, telephony, command and control systems, distributed interactive simulation, games, RT monitoring, etc.

TCP is not suited for RT application. In a RT application, the source generates a stream of data at constant rate, one or more destinations must deliver that data to applications at the same rate.

- TCP is not suitable for multicast.
- TCP includes mechanisms for retransmission of lost segments, which then arrive out of order. Such segments are not usable in most RT applications.
- TCP contains no convenient mechanism for associating timing information with segments, which is another RT requirementTCP is not suited for RT application.

6.5.2. The Transport of Real-time Traffic

- *Delay jitter* --- maximum variation in delay experienced by packets in a single session.
- *Continuous data source* --- Fixed size packets are generated at fixed intervals, e.g., traffic control radar & RT simulations.
- *On/Off source* --- fixed-size packets are generated at fixed intervals and periods of inactively, e.g., voice source (telephony, audio conf., fits this profile).

- *Variable pkt size* --- the source generates variable-length pkts at uniform intervals, e.g., digital video in which different frames may experience different compression ratios for same output quality level.

6.5.3. Requirement for RT Communication

- Low jitter.
- Low latency.
- Ability to easily integrate non-real-time services.
- Adaptability to dynamically changing net and traffic condition.
- Good performance for large network and of connections.
- Modest buffer requirements within the network.
- High effective capacity utilisation
- Low overhead in header bits per packet
- Low processing overhead per packet within network at end system.

6.5.4. RTP Protocol Architecture

Without the application-specific information, RTP is not a full protocol. On the other hand, RTP imposes a structure and defines common functions so that RT applications are relieved of part of their burden.

Application Level Framing

Two scenarios in which it might be more appropriate for recovery from lost data to be performed by the application:

- Applications may, within limits, accept less than perfect delivery and continue unchecked. This is RT audio and video. If too much data are lost, source might move to a lower-quality transmission that place lower demands on network, increasing probability of delivery.
- It may be preferable to have application rather than transport protocol provides data for retransmission. It is useful because:
  a) The sending application may recompute lost data rather than store them.
  b) Sending application can provide revised values rather than simply retransmitting lost values.

6.5.5. RTP Data Transfer Protocol

RTP supports the transfer of real-time data among a number of participants in a session. A session is defined by:

- RTP port number: the destination port, address is used for all RTP participates.

- RTCP port number: destination port address is usd by all participants for RTCP transfers.
- IP addresses: Either unicast or multicast IP.

The RTP data transfer protocol is used for the transmission of user data, typically in multicast fashion among all participants in a session.

Each RTP packet includes a fixed header and may also include additional application-specific header fields.

RTP Control Protocol

The RTP control protocol (RTCP) also operates in a multicast fashion to provide feedback to RTP data sources as well as all session participants. Four functions are performed by RTCP:

- QoS and congestion control.
- Identification
- Session size estimation and scaling.
- Session control.

6.6. Review Questions

1. Briefly explain the operation of TCP.
2. How does the window size of TCP affect the performance?
3. Briefly explain the congestion control strategies in TCP-based Internet.
4. Briefly explain the RTP.

7. IPv6: The Next Generation of Internet

7.1. Study Points

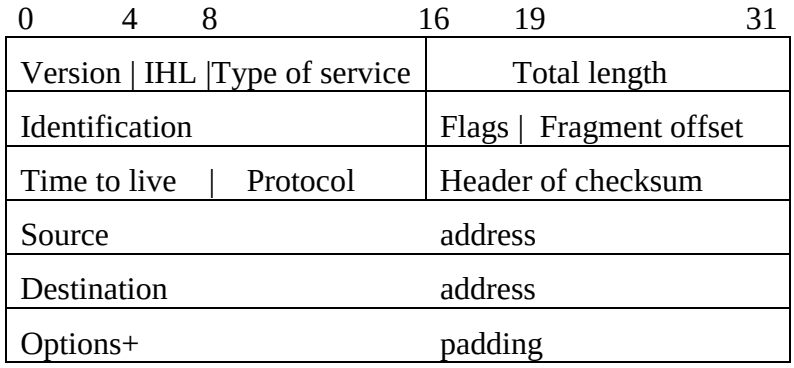
- Understand the need for inter-network traffic management.
- Understand the current Internet protocol (IPv4).
- Understand the need for the next generation of IP (IPv6).
- Be familiar with the characteristics of IPv6.
- Be familiar with the strategies for deploying IPv6.

Reference: [STA98] Chapter 11 and Chapter 12.

7.2. The current Internet Protocol

7.2.1. IPv4

IPv4 is currently the standard IP used in TCP/IP. The IPv4 IP format is:



- Version (4 bits) = 4.
- IHL = Length of header in 32-bits words (4 bits).
- Type of services: provides guidance to end-system IP modules and to routers along datagram's path.
- Total length = total datagram length in octets.
- Identifier (16 bits) = (number of segments, source address, destination address, user protocol).
- Flags (3 bits): Only 2 of the bits are currently defined. The *More* bit indicates whether this is the last fragment in original datagram, the *Don't Fragment* bit prohibits fragmentation when set. The datagram will be discarded if it exceeds the maximum size of an en route subnet. When set, it is advisable to use source routing to avoid subnet with small maximum packet size.

- Fragment Offset (13 bits): Indicates whether in original datagram this fragment belongs, measured in 64-bit units.
- Time to Live (8 bits): specifies how long, in seconds, the datagram is allowed to remain in Internet. Every routers that processes a datagram must decrease TTL by at least one, TTL ←→ hop count.
- Protocol (8 bits): indicates the next higher-level protocol, which is to receive the data field at the destination.
- Header Checksum (16 bits).
- Source Address (32 bits); Destination address (32 bits).
- Options (variable): Encodes the options requested by sending user.
- Padding (varible): Used to ensure that the datagram header is a multiple of 32-bits in length.
- Data = (max) 65,535 octets.

7.2.2. Type of Service (TOS)

The TOS definition:

TOS = (Precedence, TOS, 0)

TOS subfield values:

- Minimum delay
- 0100 Maximum throughput
- Maximum reliability
- 0001 Minimum monetary cost
- 0000 normal service

In practice, routers may ignore this subfield. If a router implements a TOS capability, there are 3 possible ways in which the routers can respond to TOS value.

- Route selection: A routing decision could be made on the basis of type of service, e.g., any datagram requesting minimum delay should not be routed through a subnet that includes a satellite link.
- Subnet service: For the next hop, the router can request a type of service from subnet that most close matches the requested TOS, some net such as ATM supports some sort of type of service.
- Queuing discipline: A router may allow TOS and precedence to affect how queues are handled. (e.g., a router may bias to min delay datagram or a route may attempt to avoid discarding datagrams that have requested max. reliability).

RFC 1349, 1812 define TOS with influence on routing decisions but did not address how the routing is made. There are general two possibilities:

(1) within a routing domain, administrator may preconfigure TOS to be associated with different routers

(2) a routing protocol could dynamically monitor TOS along various routes by monitoring delay, throughput, and dropped datagram such as OSPF.

RFC 1812 specifies the following rule for forwarding a datagram

- The router determines all available routes to destination; if there are none, the datagram is discarded.
- One or more routes have the same TOS, router chooses the route with best metric based on routing algorithm.
- Otherwise, one or more TOS = 0 (normal service), the best of these routes is chosen.
- Otherwise, the router discards the datagram.

Precedence subfield values

- 111 Network Control
- 110 Internetwork control: router-based control message
- 101 Critical
- 100 Flash override
- 011 Flash
- 010 Immediate
- 001 Priority
- 000 Routine

This subfield is set to indicate the degree of urgency or priority to be associated with datagram.

RFC 1812 specifies the following queue service for forwarding a datagram

a. Router should implement precedence-ordered queue service → when a pkt is selected for output on a logical link, the pkt of highest precedence that has been queued for that link is sent.

b. Any router MAY implement other policy-based throughput management that result in other than strict precedence ordering, but it MUST be configurable to suppress them.

Congestion control. When a router receives a packet beyond its storage,

a. It may discard the pkt just received, but this is not a good solution.

b. A router should select a packet from one of the sessions most heavily abusing the link. A recommended policy in datagram environment using FIFO queues is to discard a packet randomly selected from the queue or fair queue. Fairer policies can be the longest queue etc.

c. Must not discard a packet with high IP precedence.

d. A router may protect packets whose IP header requests the maximum reliability TOS, except for doing so may violate previous rules.

e. A router may protect fragmented IP packets, on the theory that dropping of a fragment of a datagram may increase congestion by causing all flags to be retransmitted by the source.

f. (Dedicated) router may protect packets with itself as the source or destinations.

7.2.3. IPv4 Options

- Security: allows a security label to be attached to a datagram.
- Source routing: maybe strict (only identified router maybe visited) or loose (other intermediate routers may be visited).
- Route recording: A field is allocated to record the sequence of routers visited by the datagram.
- Timestamping: The source IP entity and some or all-intermediate routers add a timestamp (precision to milliseconds) to data unit as it goes by.

7.3. IPv6 Introduction

TCP/IP is expected to migrate from IPv4 to IPv6 but it will take many years to complete.

7.3.1. Why IPv6?

The primary motivation for change from IPv4 to IPv6 is the limited address space. The 32-bit IP address can only include over a million networks in the Internet. At the current growth rate, each of the possible network prefixes will soon be assigned and no further growth will be possible.

The second motivation for change comes from requirements of new applications, especially applications that require real-time delivery of audio and video data. The current IP has limited capabilities for routing real-time data.

Challenges faced by IPv4 can be summarized:

- Address spaces
    - growth of the Internet. Maximum: 4 billion
    - when will addresses run out? Estimates: 2005
    - single IP addresses for devices
- Mobile Internet
    - Internet services from everywhere
    - Removing location dependency
- Security
    - End-to-end encryption

Data Integrity, authentication Requirements for the new protocol can be summarized as follows:

- Support billions of hosts
- Reduce size of routing tables
- Simplify protocol, process packets faster
- Provide better security (authentication & privacy)
- Better QoS (particularly for real-time data)
- Aid multicasting, anycasting
- Make it possible for a host to roam without changing its address

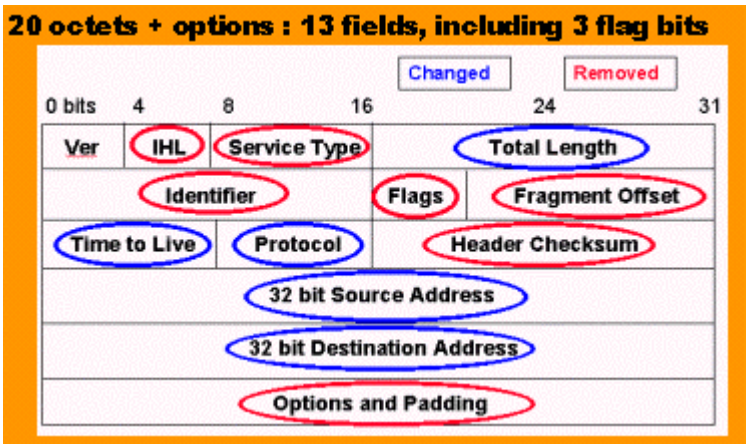


Figure 7.1. What has been changed in an IPv4 header.

7.3.2. IPv6 Features

IPv6 retains many design features of IPv4. It's connectionless. IPv6 has the following new features:

- Address size: Instead of 32, each IPv6 address is 128 bits. The address space is large enough for many years of growth of Internet.
- Header format: The IPv6 header has a completely format compared to IPv4 headers.
- Extension header: Unlike IPv4, which uses a single header format for all datagrams, IPv6 encodes information into separate headers. A datagram of IPv6 contains a base header followed by 0 or more extension headers, followed by data.
- Support for audio and video. IPv6 includes a mechanism that allows a sender and receiver o establish a high-quality path through the underlying network and to associate datagrams with that path.
- Extensible protocol. Unlike IPv4, IPv6 does not specify all possible protocol features. Instead, the designers have provided a scheme that allows a sender to add additional informaion to a datagram. The extension makes IPv6 more flexible than IPv4.

7.3.3. IPv6 Formats

(IPv6 header, extension header, …, extension header, Transport PDU)

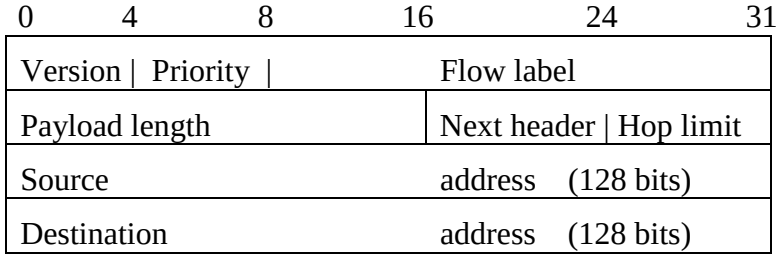
More specifically:

(IPv6 header, Hop-by-hop options header, …, Routing header, Fragment header, Authentication header, Encapsulating security payload header, Destination options header, TCP header, application data)

Extension headers:

- Hop-by-hop options header: defines special options that require Hop-by-hop processing.
- Routing header: Provides extended routing, similar to IPv4 source routing.
- Fragment header: contains fragmentation and reassembly information.
- Authentication header: provides packet integrity and authentication.
- Encapsulating security payload header: provides privacy.
- Destination options header: for options to be processed only by the final destination of the packet.

7.3.4. IPv6 Header



Priority Field: enables a source to identify the desired transmit and delivery priority of each packet relative to other packets from same source no matter either the source is provided with congestion control or not.

Congestion-controlled-traffic → source "backs off" in response to congestion.

IPv6 defines the following categories of congestion-controlled traffic in decreasing order:

- Internet control traffic: It is most important traffic to deliver, especially during the time of high congestion, -- e.g, OSPF and BGP (board gateway protocol) need to receive updates concerning traffic conditions so as they can adjust routes to try to relieve congestion. SNMP need the report congestion to applications and be able to perform dynamic reconfiguration and to alter performance-related parameters to cope with congestion.
- Interactive traffic: such as on-line user-to-host connections. User efficiency is critically dependent on rapid response time during interactive sessions. Delay must be minimised.

- Attended bulk transfer: such as FTP or HTTP (hypertext transfer protocol), HTTP may not do congestion control).
- Unattended data transfer: User will not wait, e,g email.
- Filler traffic: expected to be handled in the background when other forms of traffic have been delivered (such as USENET).
- Uncharacterised traffic: upper-layer (no priority) traffic.
- Non-congestion-controlled traffic – constant data rate and delivery delay. Examples: real-time video/audio.

No relationship between congestion-controlled and non-congestion control traffics.

Headers in an Ipv6 packet are chained together as shown in Figure 7.2.

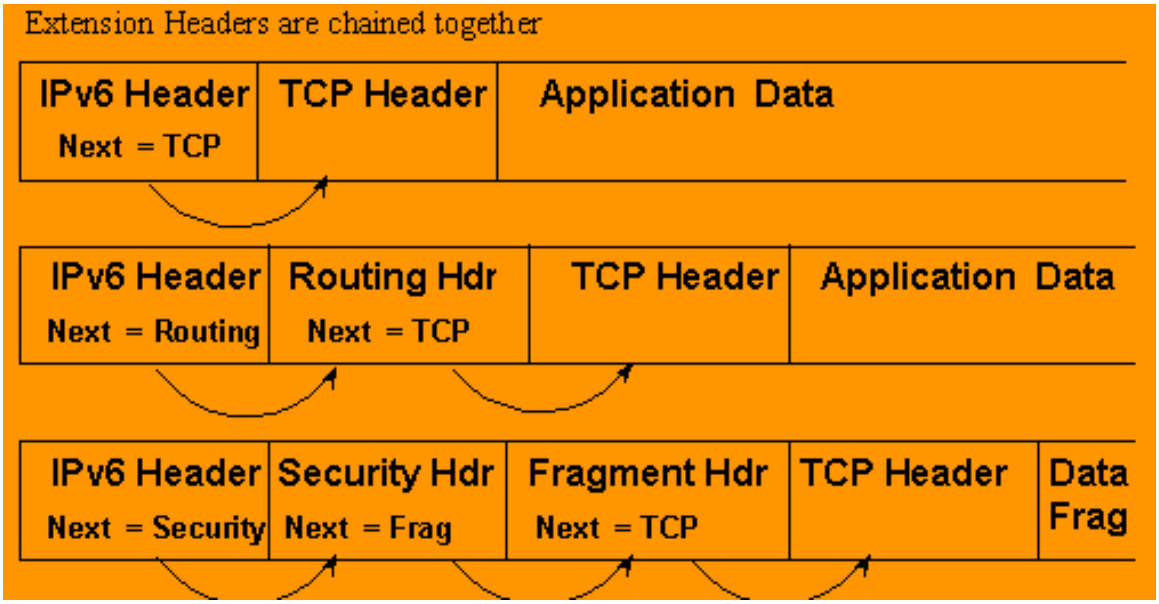


Figure 7.2. Extension headers are chained together

7.3.5. Flow Label

Ipv6 defines a flow as a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.

A flow is uniquely identified by the combination of a source address and nonzero 24-bit flow label. All packets that are to be part of the same flow are assigned the same flow label by the source.

A flow may comprise one TCP or multiple TCP connections, e.g., for multimedia conference: one for audio and one for graphic window each with different QoS requirements of data rate, delay and delay variation.

Router's Point of View: how they handle the flow such as path, resource allocation, discard requirements, accounting and security attributes.

Different flows may be treated by routers with different queue (buffer) assignments, precedence, QoS requirements for subnets.

There is no special significance to a particular flow. Applications may negotiate with network for the connection control protocol (algorithm) for any special treatment such as non-default QoS and real-time service.

A router has to save the flow requirement information about each flow.

The following rules apply to the flow label (FLL):

- Hosts / routers that do not support the FLL field must set the field to zero when originating a packet, pass the field unchanged when forwarding a pkt, ignore the field when receiving a packet.
- All packets originating from a given source with the nonzero FLL must have the same *Destination Address*, *Source Address*, *Priority*, *Hop-by-hop* options header. Intent is to allow a router to look into the flow label in a table without examining the rest of header.
- A source assigns randomly in the range 1 to $2^{24} - 1$, subject to the restriction that a source must not reuse a FLL for a new flow within the lifetime of existing flow.
- Most routers use a hashing function on flow label, which may be low-order few bits of the FLL or some simple calculation on the 24-bits of FLL.

7.3.6. IPv6 Addresses

Longer Internet addresses allow for aggregating addresses by hierarchies of network, access provider, geography, corporation and so on.

IPv6 allows three types of addresses:

- Unicast: an identifier for a single interface.
- Anycast: an identifier for a set of interfaces. A packet sent to an anycast address is delivered to one of the interfaces.
- Multicast: an identifier for a set of interfaces. A packet sent to a multicast address is delivered to all interfaces.

7.3.7. Hop-by-Hop Options Header

If this header presents, it must be examined by every router along the path. The header consists of the following:

- Next Header (8 bits): Identifies the type of header immediately following this header.
- Header Extension Length (8 bits): Length of this header in 64-bit units, not including the first 64 bits.
- Options: A variable-length field consisting of one or more option definitions – 32-bits and gives payload length in octets. Allows Jumbo Payload (2^{16-1}=65535 octets). IPv6 header must be set to zero. Allows IPv6 to make the best use of available capacity over transmission medium.

7.3.8. Other Headers

The *fragment header* is only for source node.

The *routing header* contains a list of one or more intermediate nodes to be visited on the way to a packet's destination. If a router does not recognise the routing type, it must discarded the packet.

The *destination options header* contains optional information and is examined only by packet's destination node.

Why does IPv6 use separate extension headers? There are two reasons:

- Economy: Partitioning the datagram functionality into separate headers is economical because it saves space. Having separate headers in IPv6 makes it possibleto define a large set of features without requiring each datagram header to have at least one field for each feature.
- Extensibility: When adding a new feature, existing IPv6 protocol headers can remain unchanged. A new *next header* type can be defined as well as a new header format.
- The chief advantage of placing new funcitnality in a new header lies in the ability to experiment with a new feature before changing all computers in the Internet.

7.3.9. Discussion on IPv6

Why does IPv6 use separate extension headers?

- Economy: Partitioning the datagram functionality into separate headers is economical because it saves space. Having separate headers in IPv6 makes it possibleto define a large set of features without requiring each datagram header to have at least one field for each feature.
- Extensibility: When adding a new feature, existing IPv6 protocol headers can remain unchanged. A new *next header* type can be defined as well as a new header format.
- The chief advantage of placing new funcitnality in a new header lies in the ability to experiment with a new feature before changing all computers in the Internet.

IPv6 characteristics

- Practically unlimited address space
- Simplification of packet header
- Optional header fields (better support for options)
- Authentication and Privacy (more Security)
- More attention to type of service

Plug & Play - Better Configuration optionsAdditional features of IPv6:

- Network management: auto configuration
    - Plug-and-Play.

- Automate network address renumbering
- DHCP support is mandated: Every host can download their network configurations from a server at startup time
- Address changes are automated
  - Stateless ; Routers advertise prefixes that identify the subnet(s) associated with a link ; Hosts generate an “interface token” that uniquely identifies an interface on a subnet ; An address is formed by combining the two.
  - Stateful ; Clients obtain address and / or configuration from a DHCP server ; DHCP server maintains the database and has a tight control over address assignments.

- Mobile IPv6:
  - IPv6 Mobility is based on core features of IPv6
  - The base IPv6 was designed to support Mobility
  - Mobility is not an “Add-on” features
    - All IPv6 Networks are IPv6-Mobile Ready
    - All IPv6 nodes are IPv6-Mobile Ready
    - All IPv6 LANs / Subnets are IPv6 Mobile Ready
  - IPv6 Neighbor Discovery and Address: Autoconfiguration allow hosts to operate in any location without any special supportNo single point of failure (Home Agent)
  - More Scalable : Better Performance
    - Less traffic through Home Link
    - Less redirection / re-routing (Traffic Optimisation)IPv6 Security:
- Security features are standardized and mandated. All implementations must offer them
  - No Change to applications
- Authentication (Packet signing)
- Encryption (Data Confidentiality)
- End-to-End security Model
  - Protects DHCP
  - Protects DNS
  - Protects IPv6 Mobility
  - Protects End-to-End traffic over IPv4 networks

7.4. IPv6 Projects

7.4.1. Existing Industry Implementations

- **Unix**: AIX, NetBSD, FreeBSD, OpenBSD, Tru64, Solaris 7, HP-UX
- **Windows**: Novell, Trumpet, Musica, Windows 2000
- **Others**: OS/360, Mac OS X
- **Routers**: Cisco, Hitachi, Notel, Nokia, Ericsson7.4.2. Experiments
- 6REN/6TAP
- 6Bone
- vBNS IPv6
- TAHI
- KAME
- NTT v6net
- RNRT
- 6INIT
- LONG
- ...

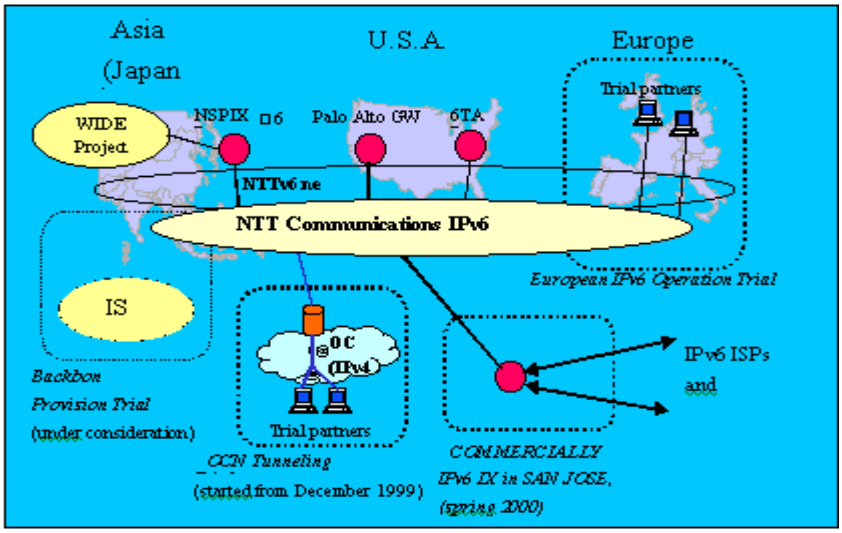


Figure 7.3. The NTT experiment

7.4.3. Depoloyment of IPv6

There are several ways to deploy IPv6 over the Internet.

- IPv6 over IPv4 TunnelsIPv6 over IPv4 Internet (Figure 7.4):

- ala 6Bone
- Any Cisco IOS routers running IPv6 software can be used at the Edge
  - 6over4 Tunnel
  - 6to4 Tunnel
- Leveraging defined Tunneling Technology

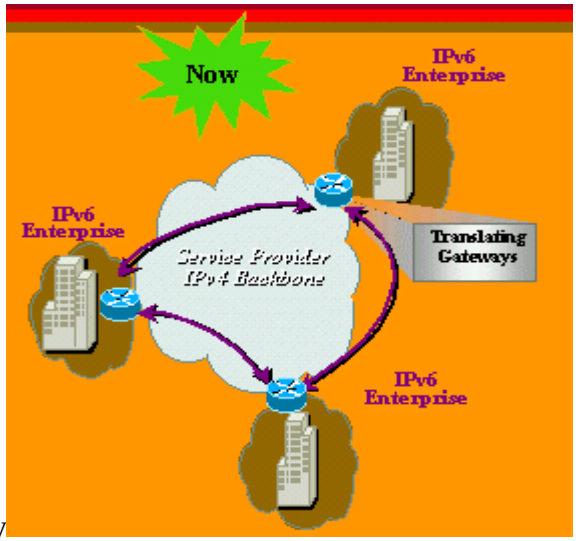


Figure 7.4. Deployment: IPv6 over IPv4 tunnels

- IPv6 over ATM & FR Backbone (Figure 7.5):
  - IPv6 over ATM or Frame Relay WAN & MAN backbones
    - Cisco BPX, MGX, IGX switches
    - Catalyst 8500MSR, LightStream 1010
  - Any Cisco routers with ATM uplink or Serial port (FR) running IPv6 software can be used

- ATM & FR PVC's running Native IPv6

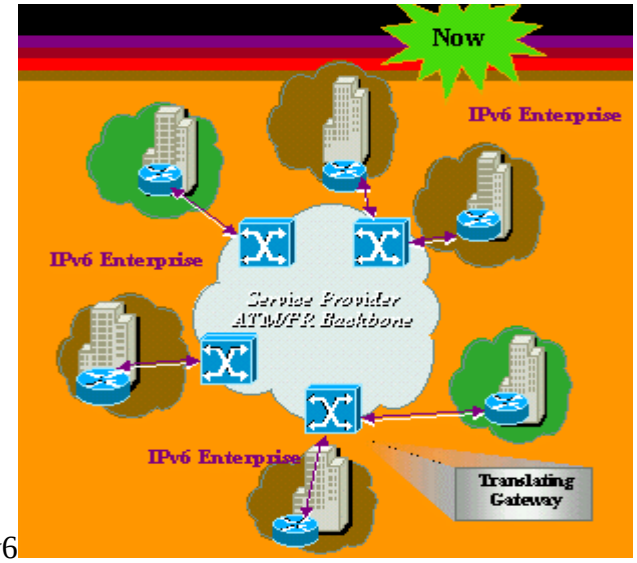


Figure 7.5. IPv6 over ATM

- Native IPv6 backbone (Figure 7.6).

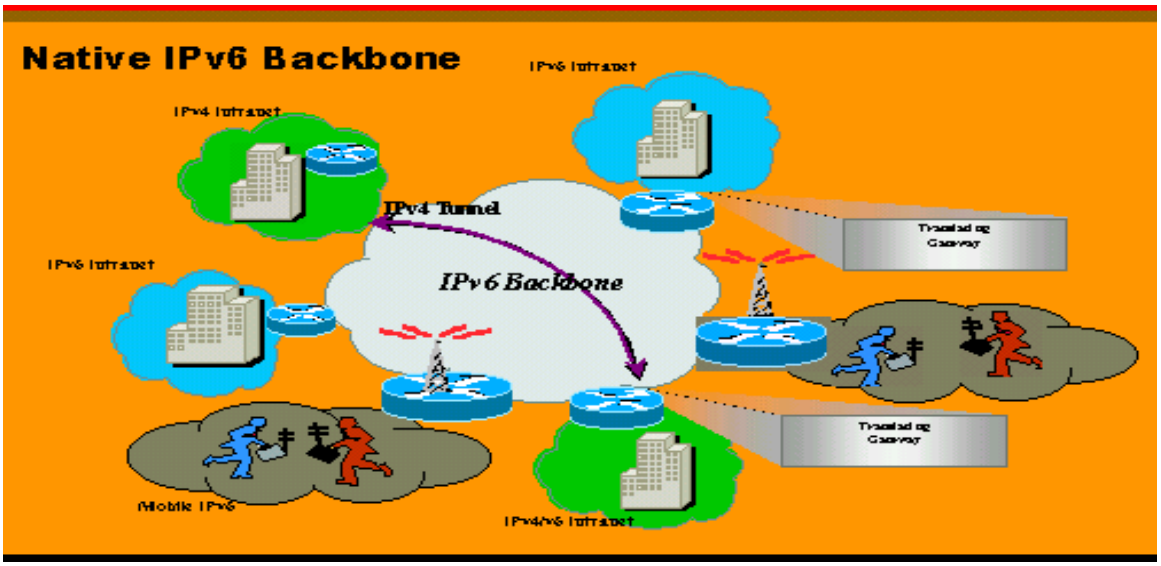


Figure 7.6. IPv6 backbone.

7.5. Other Projects and References

NASA' Interplanetary Project

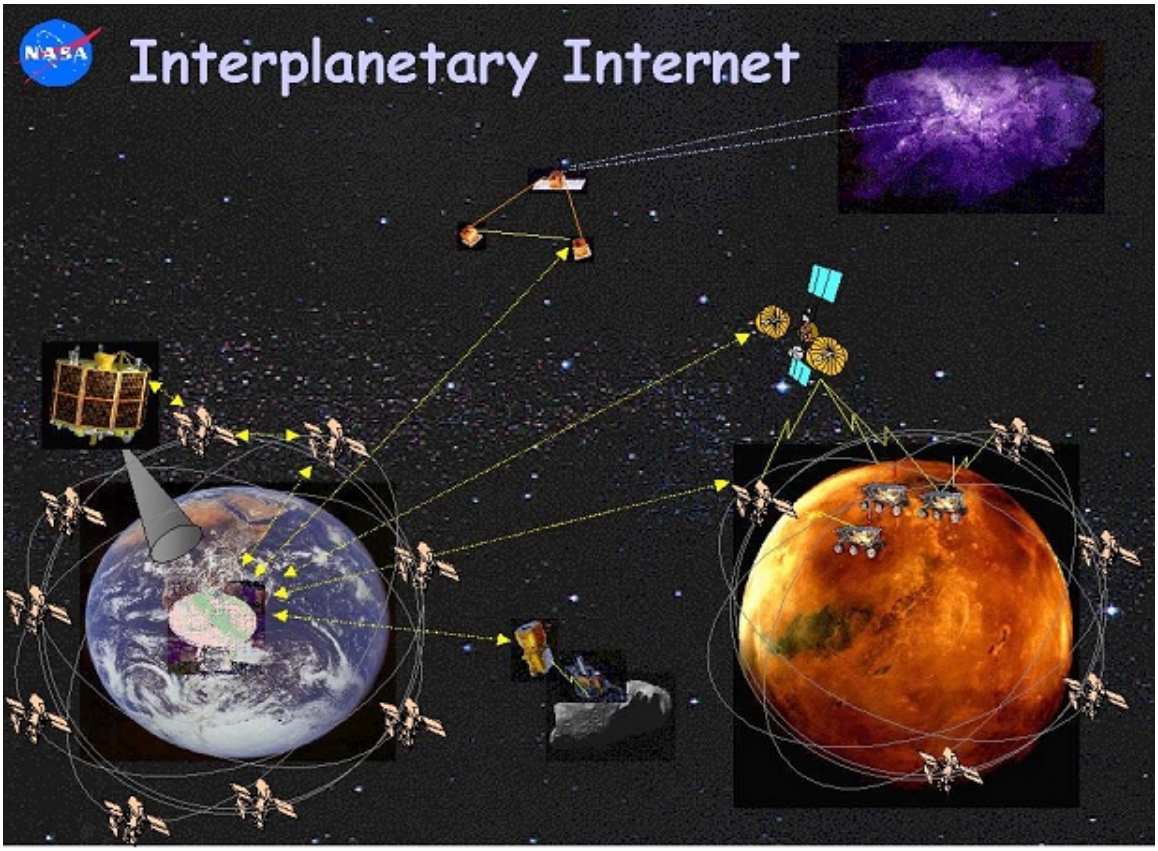


Figure7.7. NASA's project

Interesting Web sites:

- http://6ren.net/.http://www.ipv6forum.com/.http://www.ipv6.org/.http://www.ipv6.com/.http://www.6bone.net/.http://www.6tap.net/.http://www.6init.org/.http://www.ipv6.euronet.be/.http://www.nas.nasa.gov/Groups/LAN/IPv6/.http://playground.sun.com/pub/ipng/html/.http://www.digital.com/info/ipv6/.http://www-nrc.nokia.com/ipv6/ipv6/index.html.

7.6. Review Questions

1. What are the major differences and similarities of IPv4 and IPv6?
2. What are the major characteristics of IPv6?
3. What is the purpose of having multiple headers in IPv6 and how does IPv6 handle multiple headers?
4. What are the strategies to deploy IPv6?

8. Routing for High Speed and Multimedia Traffic: Multicasting and RSVP

8.1. Study Points

- Understand the routing principles in high-speed internets.
- Understand the principles of the four routing protocols: RIP, OSPF, BGP, and IDRP.
- Understand the requirements for multimedia traffic routing.
- Understand the routing protocols needed for supporting multicasting.
- Understand the techniques for users to reserve network capacity.

Reference: [STA98] Chapter 14 and Chapter 15.

8.2. Internet Routing Principles

One of the most complex and critical aspects of interent design is routing since internets operate on the basis that routers forward IP datagrams from the source to the destination.

8.2.1. The Routing Function

When receiving and forwarding IP datagrams, routers need to make routing decisions based on knowledge of the topology and conditions of the internet.

The routing decision is based on some form of least-cost criterion, e.g., to minimise the number of hops. Costs can be associated to each hop, e.g., the cost can be inversely proportional to the link capacity, proportional to the current load on the link, or some combination.

Two types of routing can be identified:

- Fixed routing.
- Adaptive routing.

In fixed routing, a simple, permanent route is configured for each source-destination pair of nodes in the network. The route are fixed, or only change when there is a change in the topology of the network. The cost of routing can be then based on the estimated traffic volumes between various source-destination pairs or the capacity of each link.

An example of the implementation of fixed routing is that, each router maintains a table that has an entry for each network and the next router to take for reaching the network. Routing decisions can be made using these entries.

In adaptive routing, as conditions in the internet change, the routers used for forwarding datagrams may change. The principal conditions influencing routing decisions:

- Failures of a network or a router.
- Congestion of a part of the internet.

Limitations of adaptive routing:

- Complexity and more burden on routers.
- Overhead of information exchanges among routers.
- The reaction speed of an adaptive strategy may cause congestion oscillation or irrelevant.
- Can produce pathologies, such as fluttering and looping.

Despite these limitations, adaptive routing strategies are by far more prevalent than fixed routing because:

(1) An adaptive routing strategy can improve performance.

(2) An adaptive routing strategy can aid in congestion control.

Classification of adaptive routing strategies: Based on information source.

- Based on local information: rarely used.
- Based on information from adjacent routers: called distance-vector algorithms.
- Based on information from routers: called link-state algorithms.

8.2.2. Autonomous Systems

An autonomous system (AS) has following characteristics:

- It consists of a group of routers exchanging information via a common routing protocol.
- It is a set of routers and networks managed by a single organisation.
- Except for times of failure, an AS is connected.

Routing protocols used for ASs are:

- An interior routing protocol (IRP) passes routing information between routers within an AS. Important examples are RIP and OSPF.
- An exterior routing protocol IERP) is used to pass information between routers of different ASs. Important examples are BGP and IDRP.

8.3. Interior Routing Protocols

8.3.1. Distance-Vector Protocol: RIP

Routing Information Protocol (RIP) is a relatively simple interior routing protocol and uses a technique called distance-vector routing. It is suitable for smaller internets and is commonly used.

Distance-vector routing requires that each node exchange information with its neighbouring nodes. Each node maintains three vectors:

- A cost vector: each element of the vector represents a link cost associated with the link.
- A distance vector: each element of the vector contains the current estimate of minimum delay from the node to a network.
- A next-hop vector: each element of the vector contains the next router in the current minimum delay route.

Routing is based on the information presented in these three vectors. Periodically (every 30 seconds), each node exchanges its distance-vector with all its neighbours and updates its vectors using the received information. Obviously, this information exchange and propagation can take a considerable amount of time.

8.3.2. Link-State Protocol: OSPF

The Open Shortest Path First protocol (OSPF) is based on the link-state routing technique and is now the preferred interior routing protocol for TCP/IP-based internets.

When a router is initialised, it determines the link cost on each of its network interfaces. The router then advertises this set of link costs to all other routers in the internet topology, not just neighbouring routers. The router also advertises the cost to all other routers if there is a significant change in the cost. Based on this information, each router can construct the topology of the entire configuration and then calculate the shortest path to each destination network.

The link-state routing uses flooding technique to advertise link cost. Flooding works as follows: a packet is sent by its source to all its neighbours. At each router, an incoming packet is retransmitted on all outgoing links except for the link on which it arrived. To prevent incessant retransmission of packets, each node is required to remember the identity of those packets it has already retransmitted. When duplicate copies of the packet arrives, they are discarded. The flooding technique is highly robust and delivers the information quickly.

The main disadvantage of flooding is the high traffic load that it generates.

OSPF has the following routing metric (cost) schemes based on the concept of type of service (TOS):

- Normal (TOS 0): The default routing metric, assigned by the administrator.
- Minimise monetary cost (TOS 2): Actual monetary costs are assigned to network use.
- Maximise reliability (TOS 4): Based on the reliability of the internets.
- Maximise throughput (TOS 8): Based on the data rate of the interface.
- Minimise delay (TOS 16): Based on a measure of the transit time of delay through a particular hop.

8.4. Exterior Routing Protocols

The two exterior routing protocols, the Border Gateway Protocol (BGP) and the Inter-Domain Routing Protocol (IDRP) are based on a technique called path-vector routing.

8.4.1. Path-Vector Routing

The path-vector routing provides information about which networks can be reached by a given router and the autonomous systems that must be crossed to get there. The approach differs from a distance-vector algorithm in two aspects:

- The path-vector approach does not include a distance or cost estimation.
- Each block of routing information lists all of the autonomous systems visited in order to reach the destination network by this route.

The path information enables a router to perform policy routing. That is, a router may decide to avoid a particular path in order to avoid transiting a particular AS.

8.4.2. Border Gateway Protocol

The BGP is developed for use in conjunction with internets that employ the TCP/IP protocol suite. BGP has become the preferred exterior routing protocol for the Internet. BGP allows routers, called gateways in the standard, in different autonomous systems to cooperate in the exchange of routing information. The current version of BGP is known as BGP-4.

Three functional procedures are involved in BGP:

- Neighbour acquisition: It occurs when two neighbouring routers in different autonomous systems agree to exchange routing information regularly. To perform neighbour acquisition, one router sends an Open message to another. If the target router accepts the request, it returns a Keepalive message in response.
- Neighbour reachability. It is used to maintain the neighbouring relationship once is established. To do so, the two routers periodically issue Keepalive messages to each other.
- Network reachability. Each router maintains a database of the subnetworks that it can reach and the preferred route for reaching the subnetwork. An Update message is broadcast to all BGP routers if a change is made to the database.

8.4.3. Inter-Domain Routing Protocol

The Inter-Domain Routing Protocol (IDRP) is an ISO standard defined within the OSI family of protocols and has been designated for use with IPv6. However, IDRP can be used with any other internet protocol.

IDRP is based on path-vector routing and represents a superset of BGP’s functions. The key differences are:

- BGP operates over TCP, whereas IDRP operates over the internet protocol used in the configuration.
- BGP uses 16-bit autonomous system numbers. IDRP uses variable-length identifiers.
- IDRP can deal with multiple internet protocols and multiple internet address schemes.

- BGP communicatesd a path by specifying the complete list of autonomous systems that a path visits. IDRP is able to aggregate this information using the concept of routing domain confederation.

8.5. Multicasting

Multimedia applications inevitably imply for multicast transmission and therefore it calls for the techniques for multicasting over an internet.

IP address can refer to a group hosts on one or more networks. Such addresses are called multicast addresses and the act of sending a packet from a source to the members of a multicast group is called multicasting.

Applications of multicasting include multimedia, teleconferencing, database, distributed computation, and real-time workgroup.

8.5.1. Multicasting strategies:

- Broadcast: If the source does not know the location of the members of the multicast group, it can then broadcast a copy of each packet to each network in the configuration, over the least-cost route for each network
- Multiple unicast: If the source knows the location of each member of the multicast group, it can then uses a table to map a multicast address into a list of networks that contain members of that multicast group. Therefore the source need only send packets to those networks that contain members of the group.
- Multicast: it uses the following method:
    1. The least-cost path from the source to each network that includes members of the multicast group is determined. This results in a spanning tree that contains only those networks containing group members.
    2. The source transmits a single packet along the spanning tree.
    3. The packet is replicated by routers only at branch points of the spanning tree.

8.5.2. Requirements for Multicasting

The following is a list of required functions for multicasting:

- A convention is needed for identification of multicast addresses.
- Each node must translate between an IP multicast address and a list of networks that contain members of this group in order to construct a shortest-path spanning tree.
- A router must translate between an IP multicast address and a subnetwork multicast address in order to deliver a multicast IP datagram on the destination network.
- A mechanism is needed by which an individual host informs routers attached to the same network as itself of its inclusion in and exclusion from a multicast group.

- Routers must exchange two types of information: (a) which subnetworks include members of a given multicast group, and (b) information to calculate the shortest path to each network containing group members.
- A routing algorithm is needed to calculate shortest paths to all group members.
- Each router must determine multicast routing paths on the basis of both source and destination addresses.

8.5.3. Internet Group Management Protocol (IGMP)

IGMP is used by hosts and routers to exchange multicast group membership information over a LAN. All IGMP messages are transmitted in IP datagrams.

The objective of each host in using IGMP is to make itself known as a member of a group with a given multicast address to other hosts on the LAN and to all routers on the LAN.

IGMP operations:

- To join a group, a host sends an IGMP report message in an IP datagram with the multicast address of the group. All hosts that are currently members of this group will receive the message and learn of the new group member.
- To maintain a valid current list of active group addresses, a multicast router periodically issues an IGMP query message, sent in an IP datagram with an all-hosts multicast address. Each host that still wishes to remain a member of the group(s) must respond with a report message for each group to which it claims membership.

IGMP was defined for IPv4 and a separate version of IGMP, incorporated into the new version of Internet Control Message Protocol (ICMPv6) has been defined for IPv6. ICMPv6 includes all of the functionality of ICMPv4 and IGMP.

8.5.4. Multicast Extensions to Open Shortest Path First (MOSPF)

MOSPF is an enhancement to OSPF that enables the routing of IP multicast datagrams. MOSPF is designed for operating in a single autonomous system.

Operations of MOSPF:

- Each router uses IGMP to maintain a current picture of local group membership.
- Periodically, each router floods information about local group membership to all other routers.
- Each router constructs the shortest-path spanning tree from a source network to all networks containing members of the multicast group.
- When a router receives a multicast packet, it performs the following actions:
    1. If the multicast address is not recognised, discard the datagram.
    2. If the router attaches to a network containing at least one member of this group, it transmits a copy of the datagram on that network.

3. The router consults the spanning tree to determine if one or more copies of the datagram should be forwarded to other routers.

8.5.5. Protocol Independent Multicast (PIM)

PIM is designed to extract needed routing information from any unicast routing protocol and may work across mutiple ASs with a number of different unicast protocols.

PIM defines two modes of operation: dense-mode and spares-mode. The dense-mode protocol is appropriate for inter-AS multicase routing and may be viewed as a potential alternative for MOSPF. The spare-mode protocol is suited for inter-AS multicast routing.

8.6. Resource Reservation: RSVP

8.6.1. RSVP Goals and Characteristics

RSVP (Resources ReSerVation Protocol) has the following design goals:

1. Provide the ability of heterogeneous receivers to make reservations specially tailored to their own needs.
2. Deal gracefully with changes in multicast group membership.
3. Specify resource requirements in such a way that the aggregate resources reserved for a multicast group reflect the resources actually needed.
4. Enable receivers to select one source from among multiple sources transmitting to a multicast group.
5. Deal gracefully with changes in routes, automatically reestablishing the resource reservation along the new paths as long as adquate resources are available.
6. Control protocol overhead.
7. Be independent of routing protocol.

Characteristics of RSVP:

- Unicast and multicast: RSVP makes reservations for both unicast and multicast.
- Sinmplex: RSVP makes reservations for undirectional data flow.
- Receiver-initiated reservation: The receiver of a data flow initiates and maintains the resource reservation for that flow.
- Maintaining soft state in the internet: RSVP maintains a soft state at intermediate routers and leaves the responsibility for maintaining these reservation states to end users.
- Providing different reservation styles: Users are allowed to specify how reservations should be made.
- Transparent operation through non-RSVP routers: These routers will simply use a best-effort delivery technique.
- Support for IPv4 and IPv6.

A reservation request issued by a destination end system is called a *flow descriptor*, and consists of a *flowspec* and a *filter spec*. The flowspec specifies a desired quality of service and the router will transmit packets with a given set of preferences based on the current flowspecs. The filter spec defines the set of packets for which a reservation is requested.

8.6.2. RSVP Operations

Much of the complexity has to do with dealing with multicast transmission. Unicast transmission is treated as a special case.

There are three reservation styles in RSVP:

- The wild-card filter (WF) style specifies a single resource reservation to be shared by all senders to this address. A good example of the use of the WF style is for an audio teleconference with multiple sites. Typically, only one person at a time speaks, so a shared capacity can be used by all senders.
- The fixed-filter (FF) style specifies a distinct reservation for each sender and provides an explicit list of senders. A good example of the use of the FF style is for video distribution. To receive video signals simultaneously from different sources requirees a separate pipe for each of the streams.
- The shared-explicity (SE) style specifies a single resource reservation to be shared among an explicit list of senders. The SE style is appropriate for multicast applications in which there are multiple data sources but they are unlikely to transmit simultanously.

8.6.3. RSVP Protocol Mechanisms

RSVP uses two basic message types: Resv and Path. Resv messages originate at multicast group receivers and propagate upstream though the distribution tree. The Path message is used to provide upstream routing information.

Operations of the RSVP protocol from the host perspective:

(a) A receiver joins a multicast group by sending an IGMP join message to a neighbouring router.

(b) A potential sender issues a Path message to the multicast group address.

(c) ©©A receiver receives a Path message identifying a sender.

(d) Now that the receiver has reserve path information, it may starts sending Resv messages, specifying the desired flow descriptors.

(e) The Resv message propagates through the internet and is delivered to the sender.

(f) The sender starts sending data packets.

(g) The receiver starts to receive packets.

Event (a) and (b) may happen in either order. Figure 10.1 illustrates the above operations.

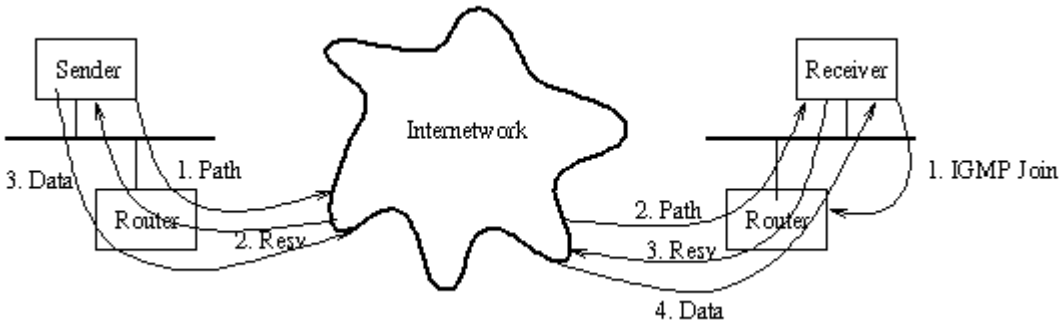


Figure 10.1 RSVP operations (host model)

8.7. Review Questions

1. What is Internet routing and why is it important?
2. Briefly explain the interior routing protocols.
3. Briefly explain the exterior routing protocols.
4. What is multicasting? How does the Internet handle multicasting?
5. What is RSVP and how does it work?

9. Client-Server Communication Using IP Sockets

9.1. Study Points

- Understand the client-server model and its applications in Internet communications.
- Understand the different types of communication services.
- Understand the principals of TCP/IP protocol and Internet Domain Sockets.
- Be able to complete simple communication programs using Internet sockets.

9.2. The Client-Server Model

9.2.1. The Basic Client-Server Model

Many applications using high-speed networks are based on the *workstation-server* model. In this model, each user is provided a single-user computer, known as a workstation. Application programs are executed in the user's workstation. File servers store and manage shared data and other specialised device servers manage expensive devices such as laser printers, plotters and scanners. One possible extension to this model is to have different types of workstations, or even some multi-user computers connected in the network, resulting in a heterogeneous computer system.

The existence of multiple, disjoint processor address spaces within a distributed system introduces the need to partition the software components to suit the hardware configuration. To accommodate this partitioning, the logical structure of the software is usually defined as a collection of cooperating modules that is then mapped onto the target hardware.

One of the widely used partition models for distributed programs is the *client-server* model. In this model, server processes manage objects and client processes access these objects by using communication facilities and operations provided by the system and servers. As servers and clients usually locate in different hosts, we call the operations provided by servers as *remote operations*.

Servers are shared by many client processes. Figure 9.1 depicts this model. Here server S_1 and server S_2 manage objects O_1 and O_2, respectively. Client C_1 and client C_2 access these objects by using the remote operations provided by the servers. These servers and clients may reside on different hosts within the network

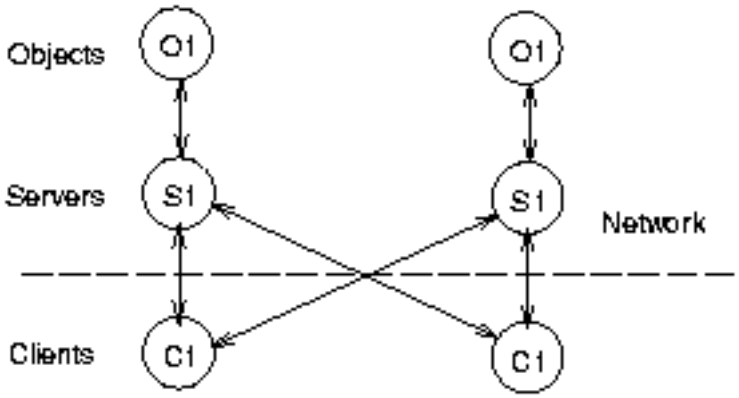


Figure 9.1. The client-server model

Two classes of communication models are frequently used in client-server communication: the *message passing* model and the *remote procedure call* (RPC) model. Program parts (clients and servers) can use the operations provided by these models to communicate with each other. In the message passing model, communications are performed by using two primitives: *send* and *receive*. One of the major advantages of the message passing model is that the *send* operation does not necessarily block the source process for waiting a response from the target process. So the interprocess communication in this model can have a lot of forms instead of just request-response pairs.

The RPC model allows a programmer to call a procedure located at a remote computer in the same manner in which a local procedure is called. One of the major disadvantages of the RPC model is the limitation of the communication form: only the request-response form is allowed, that is, the calling process is blocked (suspended) until the call completes and a reply has been received, as shown by Figure 9.2. Of course, there have been some RPC systems that also support *asynchronous remote procedure calls* (or non-blocking RPCs), but they are viewed, to some degree, as a violation of the original RPC design intention. Also, these non-blocking RPCs can be modeled by using the message passing model.

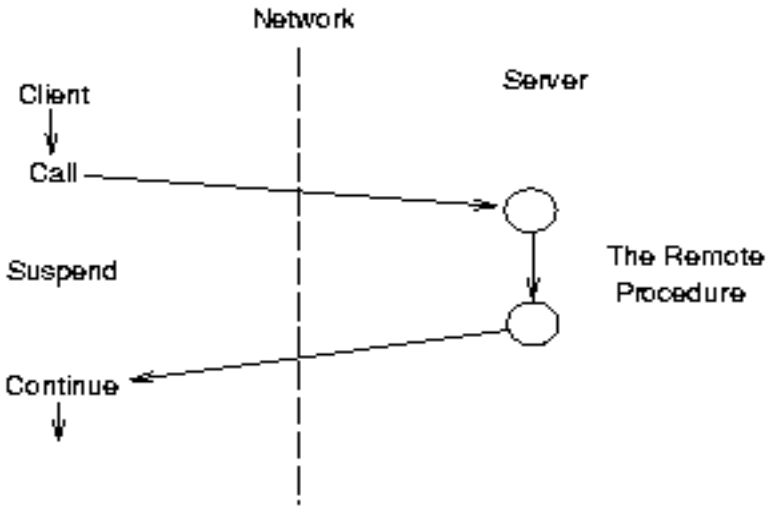


Figure 9.2. The remote procedure call communication model

The RPC model has many advantages. The procedure call is a widely accepted, used and understood abstraction. This abstraction is the sole mechanism for accessing remote services. So the interface of a remote service is easily understood by any programmer with a good knowledge ofordinary programming languages.

9.2.2. Communication Services

When talking about communications between different entities (computer hosts or programs) of a distributed software system, we can also view them as *connection-oriented* communications or *connectionless* communications. For connection-oriented communications, the following three steps are involved in the communication:

- Connection. One of the communicating entities issues a connection call and a communication path between the two communicating entities is established before the real data exchange occurs.
- Data exchange. After the connection establishment, the two communicating entities can then exchange their data in any direction. The order of data packets is preserved.
- Disconnection. After the data exchange, one of the communicating entities may issue a disconnection call and disconnect the communication path.

In connectionless communications, no connection path is required before the real data exchange between two communicating entities. Any entity wanting to send a message can send it immediately to the underlying communication system. The message may carry some information other than the real message, such as the source identifier, the destination identifier, priority, etc. One of the characteristics of connectionless communication is that the message order may be different between the sender and the recipient. Usually a connection-oriented service is more expensive than a connectionless service, because the former needs to establish, maintain, and disconnect the connection.

The connection-oriented service is modeled after the telephone system. To talk to your correspondent, you pick up the phone and dial a number (connection), then talk (data exchange), then hang up (disconnection). The essential aspect of a connection is that it acts like a tube: the sender pushes messages into the tube at one end, and the receiver takes them out from the other end in the same order.

In contrast, the connectionless service is modeled after the postal system. To send a letter (message) to your correspondent, you write the full destination address on the envelope, pack your letter into the envelop and drop it into the mail box. All these letters are routed through the postal system independently. It is possible that the letters arrive at the receiver's side in a different order to that sent.

A *reliable* communication service is one that never loses data. In this kind of service, usually an acknowledgement is sent back to the sender from the receiver indicating that a message has been correctly received. The acknowledgement introduces overhead and delay. Sometimes this is worthwhile, and sometimes this is not necessary. For example, during file transfer, the file owner wants every bit of the file to arrive at the destination in the same order as it is sent, and with no errors. In that case, a reliable connection-oriented communication service is appropriate. But if the application is digitised voice traffic, an unreliable connection-oriented service is appropriate: It is preferable for users to hear some noise on the line or lose a few words from time to time than to introduce a delay to a wait acknowledgement.

Reliable connection-oriented service has two minor variations: message sequences and byte streams. In the former, the message boundaries are preserved. This is appropriate when, for example, transferring a book over a network to a laser printer and these pages are sent as separated messages. In the latter, the connection is simply a stream of bytes, with no message boundaries. This is appropriate when, for example, the application is a remote login from a terminal to a mainframe.

A connectionless service can also be reliable or unreliable. An unreliable (meaning not acknowledged) connectionless service is often called a *datagram service*, by analogy with a telegram service, which also does not provide an acknowledgement back to the sender. An *acknowledged datagram service* provides a reliable and connectionless service. It is like sending a registered letter and requesting a return receipt. When the receipt comes back, the sender is absolutely sure that the letter was delivered to the intended correspondent.

9.3. Distributed Application Model

Figure 9.3 indicates the terms we have used to describe our distributed applications. There are two classes of distributed applications (or equally, distributed programs): message passing-oriented distributed applications and RPC-oriented distributed applications. As we have seen, an RPC-oriented distributed application can be again divided into server program parts and client program parts. A message-oriented distributed application can be further divided into sender program parts and receiver program parts. If we use message-passing tools to implement client / server distributed programs, we would have client program parts and server program parts. We shall simply call these {\it program parts (PP)}\index{program part}.

Of course, in real programming practice, a program part may act as a client as well as a server, or may act as a message sender as well as a message receiver.

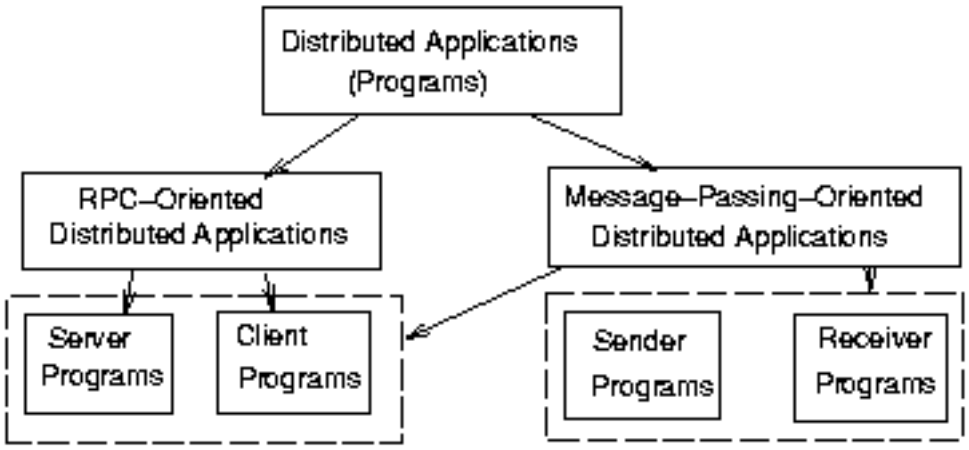


Figure 9.3. Terms used for distributed applications

One important class of distributed application is the distributed information system (IS) applications. In a distributed IS application, there is usually a number of computers and processes managing some shared information, such as databases or files. User programs access these computers and processes to obtain the information the user needs, or to update the stored information through these computers and processes. Time, in a distributed IS application, is not as critical as in a distributed real-time application. we focus on distributed IS applications here. From now on, when we mention distributed application, we mean distributed IS applications. Figure 9.4 is a generic model of a distributed application.

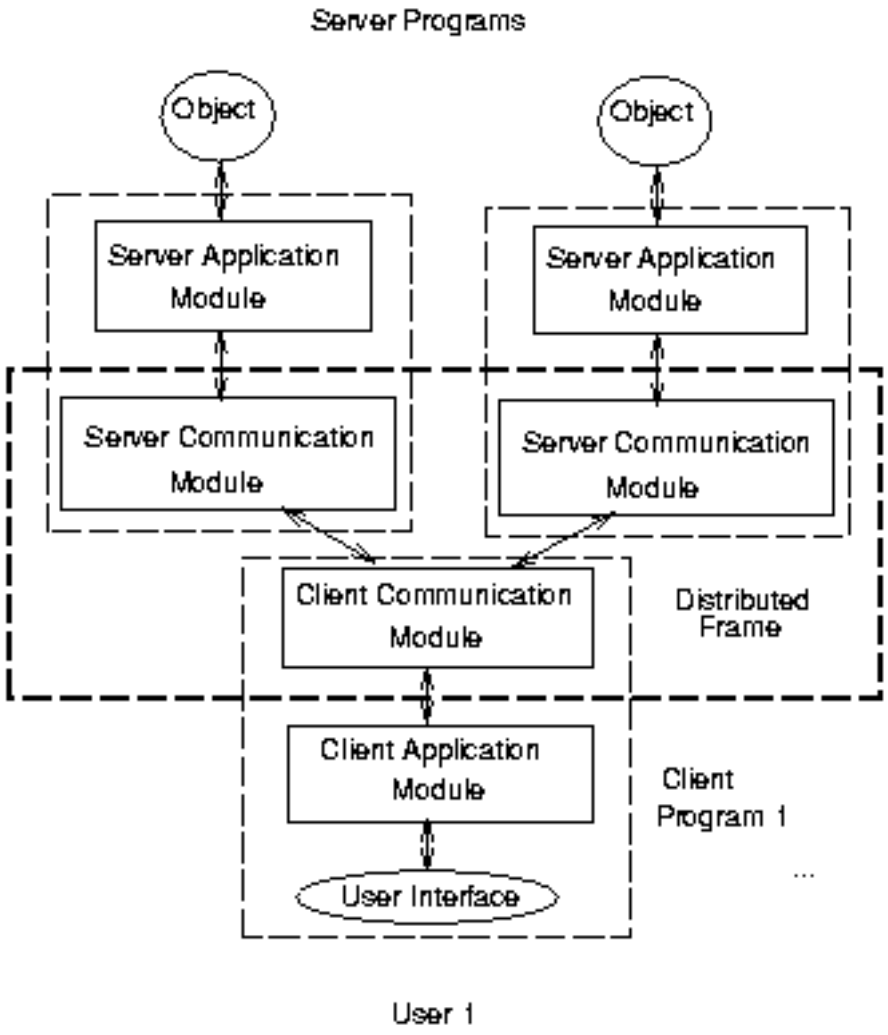


Figure 9.4. The distributed application model

According to Figure 9.4, a distributed application consists of several client programs and several server programs. Usually a server program is located on a remote computer and a client program is located on the user's (local) computer. A client program interfaces with the user, manages the local application process, and performs the communication between the client program and other related (remote) server programs. A server program usually manages an object (e.g. one part of a distributed database), performs the operations required by other programs, and manages the communications. Of course, the client program may also perform some operations directly on the local objects. This is not shown in the diagram because we want to emphasise the distributed characteristics of the application here. So, we can divide a distributed application into three parts:

- User interface. This deals with the interactions between the client program and the user.
- Distributed frame. This performs the communications among all the co-operative parts over the network.

- Application modules. They manage the objects and perform operations.

9.4. BSD Internet Domain Sockets

The ARPANET sponsored by the Advanced Research Projects Agency (ARPA) and developed during late 1960s and early 1970s is a milestone for computer networks. In the early 1980s, a new family of protocols was specified as the standard for the ARPANET. Although the accurate name for this family of protocols is the "DARPA Internet protocol suite," it is commonly referred as the TCP/IP protocol suite, or just TCP/IP.

The *Internet domain sockets* on BSD UNIX use the TCP/IP protocol suite as the communication protocols among processes generally located on different computers across a network. We introduce the TCP/IP in this section.

9.4.1. Overview

Communications between computers connected by computer networks use well-defined protocols. A protocol is a set of rules and conventions agreed by all of the communication participants. As we have mentioned, in the OSI reference model, the communication protocols are modeled in seven layers. Layered models are easier to understand and make the implementation more manageable. A *protocol suite* is defined as a collection of protocols from more than one layer that forms a basis of a useful network. This collection is also called a *protocol family*. The TCP/IP protocol suite is an example.

There are many protocols defined in the TCP/IP protocol suite. We are going to describe three of them: the Transport Control Protocol (TCP), the User Datagram Protocol (UDP) and the Internet Protocol (IP). If using OSI reference model the TCP and UDP protocols are Transport layer protocols, while the IP protocol is a Network layer protocol. Figure 9.5 illustrates the relationship of these protocols and their positions in the OSI reference model.

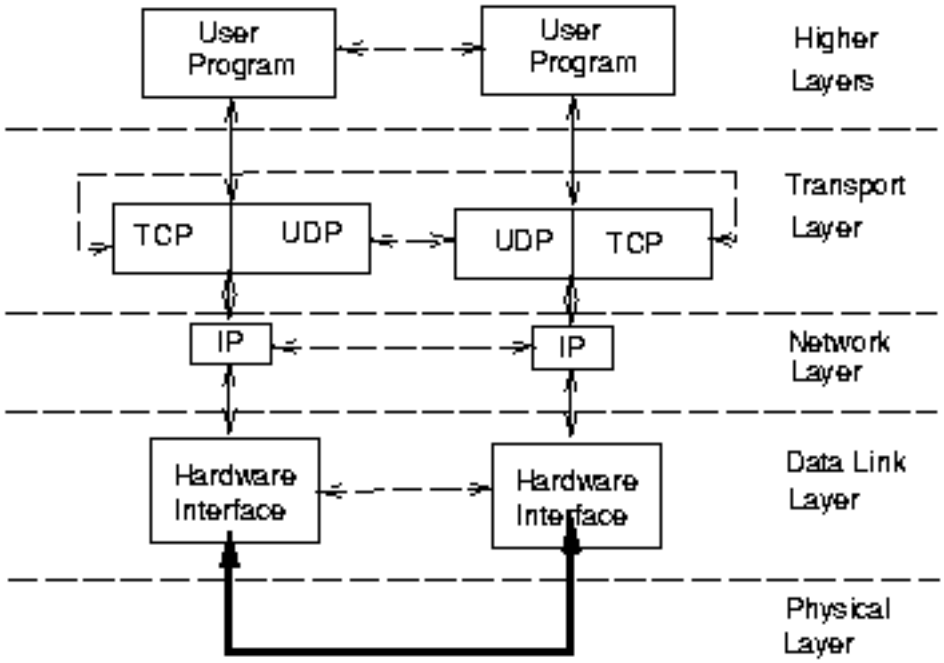


Figure 9.5. The Layered TCP/IP protocol suite

The TCP protocol is a connection-oriented protocol that provides a reliable, full-duplex, byte stream for interprocess communications. The UDP protocol, on the other hand, is a connectionless protocol that provides an unreliable datagram service: there is no guarantee that UDP datagrams ever reach their intended destination. Acknowledgement must be used in order to provide reliable services when using the UDP protocol.

9.4.2. Network Layer: IP

The IP protocol is connectionless and unreliable. It is based on *internet datagrams*. The protocol takes a datagram from the transport layer (for example, from the TCP protocol). A datagram is up to 64k bytes long and it may be part of a longer message. Each datagram is transmitted over the network independently so the communication ls connectionless. During transmission, a datagram may be lost or may be further fragmented into smaller units (called *IP packets*) as it goes through the protocol layers. When all the IP packets of a datagram finally arrive the destination computer, they are reassembled to form the datagram and then transferred to the transport layer of the destination site. If any of the IP packets of a datagram are lost or corrupted, the entire datagram is discarded by the destination site so the IP protocol is therefore unreliable because it cannot guarantee the delivery of a datagram.

The IP datagram consists of a header and a text part. The header includes information such as the type of service, the length of the header, the length of the text part, the address of the source computer, the address of the destination computer, and other information.

It is the IP layer that handles the routing through networks. The Internet address is used to identify networks and computers and is used in an IP datagram header to denote the

source and destination computer addresses. An Internet address has 32 bits and encodes both a network ID number and a host ID number. Every host on a TCP/IP Internet must have a unique Internet address. The network ID numbers are assigned by some kind of authority, the Network Information Center (NIC) located at SRI International. While the host ID numbers are assigned locally.

The common notation of an Internet address is to use 4 bytes, as shown in the following:

`field_1.field_2.field_3.field_4`

Where $0 <= field_i <= 255_{10}$ (FF_{16}), $1 <= i <= 4$.

Depending on the network's class (described below), the network number can be $field_1$, or $field_1.field_2$ or $field_1.field_2.field_3$. That means the host number can be $field_2.field_3.field_4$, $field_3.field_4$ or $field_4$.

Networks are classified into 3 classes, as listed in the next Table.

Class	Binary number of $field_1$	Network ID (decimal)
A	000 000 – 0111 111	0 - 126
B	1000 000 – 1011 1111	128 – 191.254
C	1100 0000 – 1101 1111	192 – 223.254.254

A brief description of these classes follows:

Class A: networks are the largest networks with more than 65,536 hosts. A class A network's network ID number is $field_1$.

Class B: networks are mid-size networks with host IDs ranging from 256 to 65,536. A class B network's network ID number is $field_1.field_2$.

Class C: networks are the smallest networks with up to 256 hosts A class C network's network ID number is $field_1.field_2.field_3$.

Figure 9.6 illustrates the Internet address formats of these three network classes.

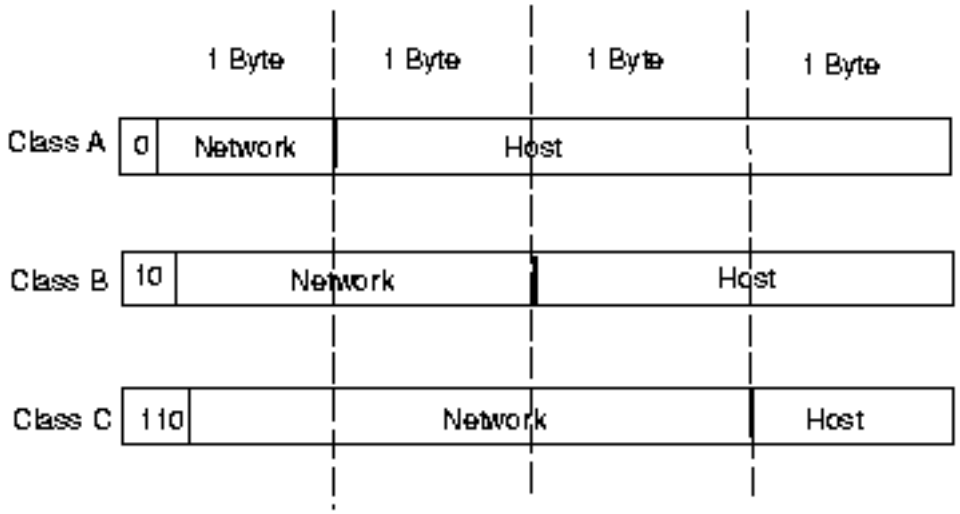


Figure 9.6. Internet network classes

For example, if we have an Internet address 98.15.12.63, we can tell that it is a class A network because $field_1$ is within the range of 0 - 126. Its network ID number is 98 and host ID number is 15.12.63. If we have an Internet address 130.194.1.106, we can tell that it is a class B network because the $field_1$.$field_2$ is within the range of 128 - 191.254. Its network ID number is 130.194 and host ID number is 1.106.

It is evident that an Internet address can only be assigned to one host. But a host can have several Internet addresses. This is because that in some situations, we want a host to be connected to several networks.

Although an Internet address clearly specifies the address of a host, few persons want to use Internet addresses directly: they are too hard to remember. Domain Name System (DNS) is used to name host addresses in more human-oriented way and to find the Internet addresses corresponding to machine names.

The DNS is a hierarchical naming system: its name space is partitioned into sub-domains, which can themselves be further divided. The DNS is also a distributed system: the name space is delegated to local sites that are responsible for maintaining their part of the database. Programs called *name servers* manage the database.

The DNS name space can be represented as a tree, with the nodes in the tree representing *domain names*. A *fully qualified domain name* is identified by the components (nodes) of the path from the domain name to the root. A component is an arbitrary string of up to 63 octets in length; the length of a fully qualified domain name is limited to 256 octets. By convention, a domain name is written as a dot-separated sequence of components, listed right to left, starting with the component closet to the root. The root is omitted from the name. Thus, *wan_res.cm.deakin.edu.au* is a fully qualified domain names. It is certainly easier to be remembered than the corresponding Internet addresses 139.130.118.102.

DNS name space is divided into *zones of authority*, and name servers have complete control of the names within their zones (domains). For easier management of domains, a large domain can be split into smaller sub-domains, and name servers can delegate authority to other name servers for sub-domains. For example, if *edu.au* represents the domain of all educational institutions in Australia, then *deakin.edu.au* and *anu.edu.au* are its two sub-domains. Queries for DNS information within sub-domain *deakin.edu.au* are first dealt with by the name server of this sub-domain. If this name server cannot answer a query, the query is then directed to the name server of *edu.au* domain. At last, the name server of the root can answer the query.

9.4.3. Transport Layer: TCP and UDP

As we have shown in Figure 9.5, user processes interact with the TCP/IP protocol suite by sending and receiving either TCP data or UDP data. To emphasise that the IP protocol is used, we sometimes refer them as the TCP/IP or UDP/IP protocols.

TCP provides a connection-oriented, reliable, full-duplex, byte-stream service, similar to a virtual circuit, to an application program. UDP, on the other hand, provides a connectionless, unreliable datagram service to an application program.

As we mentioned in the previous section, the Internet address is used to identify networks and computers. In order to let many processes use the TCP or UDP simultaneously (these

processes may reside on any computers of a network), both protocols use 16-bit integer t *port numbers* for identifying data associated with each user process. The association of port numbers and user processes last as long as the communication, so the following 5-tuple uniquely identifies a communication:

- the protocol (TCP or UDP),
- the local computer's Internet address,
- the local port number,
- the foreign computer's Internet address,
- the foreign port number.

For example, if we have a communication using TCP protocol. The server is on a host with domain name of *wan_res.cm.deakin.edu.au* (Internet address 139.130.118.102), using port number 5100. The client is on a host with domain name of *sky3.cm.deakin.edu.au* (Internet address 139.130.118.5), using port number 5101. The 5-tuple which uniquely defines the communication is:

{tcp, 139.130.118.102, 5100, 139.130.118.5, 5101}

Because the host name is easier to understand and there are some system calls to convert between a host name and its Internet address, the above 5-tuple can then be written as:

{tcp, wan_res.cm.deakin.edu.au, 5100, sky3.cm.deakin.edu.au, 5101}

Because *wan_res.cm.deakin.edu.au* and *sky3.cm.deakin.edu.au* are within the same sub-domain, we can even write the 5-tuple as:

{tcp, wan_res, 5100, sky3, 5101}

There are some restrictions in using port numbers. In TCP and UDP, port numbers in the range 1 through 255 are reserved. All well-known ports (some commonly used utilities use these ports) are in this range. For example, the File Transfer Protocol (FTP) server uses the well-known port number 21 (decimal). Some operating systems also reserve additional ports for privileged usages. For example, 4.3BSD reserves ports 1-1023 for superuser processes. Only port numbers of 1024 or greater can be assigned by user processes.

A TCP protocol entity accepts arbitrarily long messages from user processes, breaks them into datagrams of up to 64k bytes, and sends them to the IP layer. Before the real communication happens, a connection must be set up between the sender and the recipient. After the communication, the connection must be disconnected.

As the IP layer does not guarantee the proper delivery of a datagram, it is the responsibility of the transport layer to ensure that a datagram arrives at the destination properly using time-out and retransmission techniques. Also as datagrams are transmitted independently, the datagrams of a message may arrive at the destination out of order and it is also the TCP protocol's responsibility to reassemble them into the message in the proper sequence.

Each datagram submitted by the TCP to IP layer contains a TCP header and a data part. The whole TCP datagram is viewed by the IP as data only and an IP header is added to

form an IP datagram. The TCP header contains the source port number, the destination port number, the sequence number, and other information.

The TCP protocol has a well-defined service interface. There are primitives used to actively and passively initiate connection, to send and receive data, to gracefully and abruptly terminate connections, and to ask for te status of a connection.

A UDP protocol entity also accepts arbitrarily long messages from user processes, breaks them into datagrams of up to 64k bytes, and sends them o the IP layer. Unlike the TCP protocol, no connection is involved and o guarantee of delivery or sequencing. In effect, UDP is simply a user interface to IP. A header is also added into the datagram by UDP, which contains the source port number and the destination port number.

9.5. IP Socket: Basic Concepts

9.5.1. Socket Model

The socket model consists of three parts: the socket layer, the protocol layer and the device layer. Figure 9.7 displays this model.

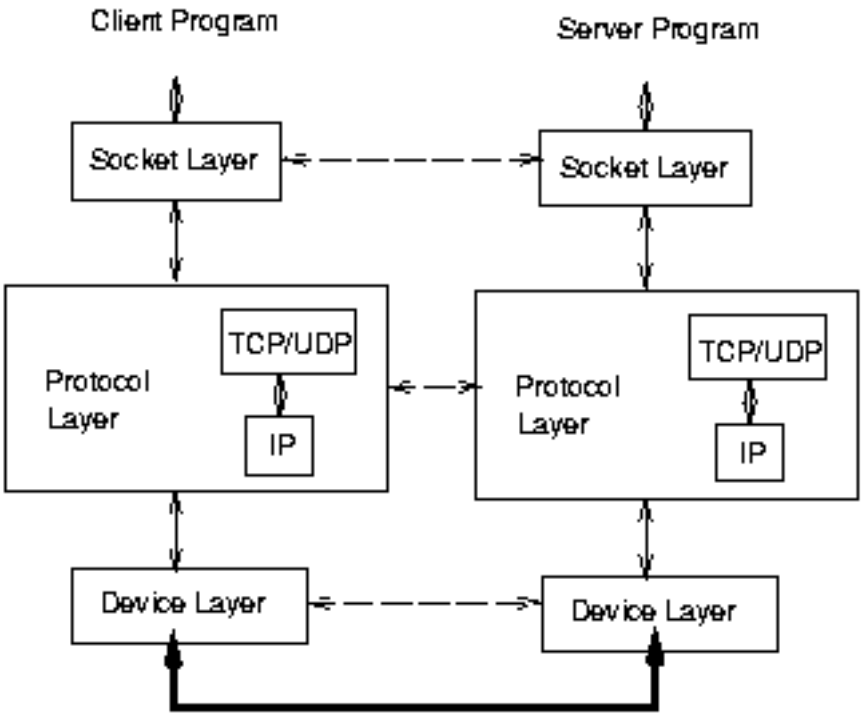


Figure 9.7. Socket Model

The layered model is designed to support the following aspects:

- Transparency: Communication between processes should not depend on whether or not the processes are on the same machine.

- Efficiency: The applicability of any interprocess communication facility is limited by its performance.
- Compatibility: Existing naive UNIX processes that read from the standard input file and write to the standard out file should be usable in a distributed environment without change.

9.5.2. Internet Domain Socket Naming

The naming facility in the Internet domain is quite complex. It is an association of local and foreign addresses, and local and foreign ports. Port numbers are allocated out of separate spaces for each Internet protocol. Associations (protocol, local address, local port, foreign address, foreign port) must always be unique for each socket.

The definition of a socket name in the Internet domain is in netinet/in.h:

```
struct in_addr {
        union {
                struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
                struct { u_short s_w1,s_w2; } S_un_w;
                u_long S_addr;
        } S_un;
#define s_addr S_un.S_addr/*can be used for most tcp & ip code*/
#define s_host S_un.S_un_b.s_b2    /* host on imp */
#define s_net  S_un.S_un_b.s_b1    /* network */
#define s_imp  S_un.S_un_w.s_w2    /* imp */
#define s_impno        S_un.S_un_b.s_b4    /* imp $ */
#define s_lh   S_un.S_un_b.s_b3    /* logical host */
};
/*
 * Socket address, internet style.
 */
struct sockaddr_in {
        short   sin_family;
        u_short         sin_port;
        struct  in_addr sin_addr;
        char    sin_zero[8];
};
```

This is quite complex, especially the *sin_addr* field. Fortunately we have some special system calls to deal with these fields and they can make the naming process much simpler. We will describe these system calls later. Actually, the name of an Internet Domain socket consists of two parts: a host name part and a port number part. As long as we know these two parts, we can use the socket.

Various types of socket addresses (e.g. the UNIX domain socket address, the Internet domain address as well as the XNS address) can be combined together into a uniform socket address, as defined in *sys/socket.h*:

```
struct sockaddr {
  u_short  sa_family;  /* address family: AF_XXX value */
  char sa_data[14];    /* up to 14 bytes of protocol-specific
                          address */
```

All the system calls that used socket address as one of their parameters actually ask for a type *sockaddr* instead of individual types (such as *sockaddr_in*). However the best way to pass a parameter to socket-related system calls is to use the cast method in the C language. For example, no matter where *my_sock* is defined as a UNIX domain socket address *sockaddr_un*, or an Internet address *sockaddr_in*, we can use the following format for a connect call:

```
connect(sockDesc, (struct sockaddr *) &my_sock, sizeof(my_sock));
```

The *sockaddr* structure definition is also the reason that in some situations in the UNIX domain only 14 characters of a path name are recognised by the system. Because only 14 characters are defined in the *sockaddr's* protocol-specific address buffer, and the address types are usually cast to *sockaddr* type, only the first 14 characters are recognised.

9.5.3. Socket Types

Sockets are typed according to the communication properties visible to a user. Properties such as reliability, ordering, and prevention of duplication of messages are determined by types. Processes are presumed to communicate only between sockets of the same type. The basic set of socket types is defined in *sys/socket.h*:

```
/*
 * Types
 */
#define     SOCK_STREAM    1          /* stream socket */
#define     SOCK_DGRAM     2          /* datagram socket */
#define     SOCK_RAW       3          /* raw-protocol interface */
#define     SOCK_RDM       4          /* reliably-delivered message */
#define     SOCK_SEQPACKET 5          /* sequenced packet stream */
```

A *stream socket* provides for the bi-directional, reliable, sequential, and unduplicated flow of data without record boundaries. It models connection-oriented virtual circuits. Sockets of type SOCK_STREAM are full-duplex byte streams, similar to pipes. A stream socket must be in a connected state before any data can be sent or received on it. A connection to another socket is created with a connect call (described later). Once connected, data can be transferred using read and write calls or some variant of the send and *recv* calls. When a session has been completed, a *close* may be performed.

The communications protocols used to implement a stream socket (SOCK_STREAM) type ensure that data is not lost or duplicated. If a piece of data for which the peer protocol has buffer space cannot be successfully transmitted within a reasonable length of time, then the connection is considered broken and calls will indicate an error with -1 returns and with ETIMEDOUT as the specific code in the global variable *errno*. The protocols optionally keep sockets "warm" by forcing transmissions roughly every minute in the absence of other activity. An error is then indicated if no response can be elicited on an otherwise idle connection for an extended period (for example, 5 minutes). A SIGPIPE signal is raised if a process sends on a broken stream; this causes processes that do not handle the signal to exit.

A *datagram socket* (SOCK_DGRAM) supports bi-directional flow of data that is not promised to be sequential, reliable, or unduplicated. A *datagram* is defined as a connectionless, unreliable message of a fixed maximum length (typically small). So a process receiving messages on a datagram socket may find duplicated messages, and possibly in an order different from the order in which it was sent. An important characteristic of a datagram socket is that record boundaries in data are preserved. No connection is required to use a datagram socket and *sendto* and *recvfrom* calls are used to send and receive datagrams. This socket type closely models the facilities found in many contemporary packet switched networks.

A *raw socket* (SOCK_RAW) is used for unprocessed access to internal network layers. It has no specific semantics. These sockets are normally datagram-oriented. But their exact characteristics depend on the interface provided by the protocol. They have been provided mainly for further development and are now available only to the super-user.

The *sequenced packet stream socket* (SOCK_SEQPACKET) is similar to datagram socket except that data are guaranteed to be received in the sequence that they are sent. These socket type also guarantees error-free data exchange.

The *reliably-delivered message socket* ({SOCK_RDM) is planned but not yet implemented.

As we usually only use stream and datagram sockets, we will not describe other socket types in this study guide.

9.6. Basic Internet Domain Socket System Calls

9.6.1. Some Special Functions

A number of functions have been provided by BSD UNIX for using sockets more easily. We are going to introduce some of them in this section.

When dealing with socket addresses, we usually need to do some operations on bit and byte strings. The following three functions are provided for this purpose:

```
bcopy(b1, b2, length)
char *b1, *b2;
int length;

bcmp(b1, b2, length)
char *b1, *b2;
int length;

bzero(b1, length)
char *b1;
int length;
```

These functions do not check the null byte for the end of a string, as is normally done in string operations. Instead, strings used here are treated as bit and byte strings. The *bcopy* function copies length bytes from string b1 to the string b2. The *bzero* function places length 0 bytes in the string b1. The *bcmp* function compares byte string b1 against byte

string b2. Both strings are length bytes long, if they are identical a zero is returned, otherwise a nonzero value is returned.

Different computer architectures may have different byte orders. Byte orders are important when expressing Internet addresses, so the following functions are provided to convert values between host and network byte orders:

```
#include <sys/types.h>
#include <netinet/in.h>

netlong = htonl(hostlong);
u_long netlong, hostlong;

netshort = htons(hostshort);
u_short netshort, hostshort;

hostlong = ntohl(netlong);
u_long hostlong, netlong;

hostshort = ntohs(netshort);
u_short hostshort, netshort;
```

These functions convert 16-bit (short integer) and 32-bit (long integer) quantities between network byte order and host byte order. *htonl* is used to convert host-to-network, in long integer; *htons* is used to convert host-to-network, in short integer; *ntohl* is used to convert network-to-host, in long integer; *ntohs* is used to convert network-to-host, in short integer;

Sometimes it is very important to know on which host our program is executing. The following system call is used to obtain the local host name:

```
gethostname(name, namelen)
char *name;
int namelen;
```

The *gethostname* returns the standard host name for the current processor in name, such as sky3.cm.deakin.edu.au. The parameter *namelen* specifies the size of the name array. The returned name is null-terminated unless insufficient space is provided. If the call succeeds, a value of 0 is returned. If the call fails, then a value of -1 is returned and an error code is placed in the global location *errno*.

As we mentioned in Section 9.4.3, a 5-tuple uniquely defines a communication:

{protocol, address(local), port(local), address(foreign), port(foreign)

Where addresses are hosts' Internet addresses. A data structure *hostent* is defined in the *netdb.h* header file containing host information. It is defined as:

```
struct   hostent {
  char   h_name;       / official name of host */
  char   **h_aliases;   /* alias list */
  int    h_addrtype;    /* address type */
  int    h_length;      /* length of address */
  char   **h_addr_list; /* list of addresses from name server */
#define  h_addr h_addr_list[0] /* address for backward compatibility */
};
```

The *h_name* field is the host's name string, the same as we obtained from *gethostname* call. The *h_aliases* field is a zero terminated array of alternate names for the host. The *h_addrtype* field currently is always AF_INET. The *h_length* field gives the length of the address (in bytes). The *h_addr*_list field is a pointer to the network address for the host. And the *h_addr* field is the host's first network address.

The following functions are used to get host information:

```
#include <netdb.h>
struct hostent *gethostbyname(name)
char *name;

struct hostent *gethostbyaddr(addr, len, type)
char *addr; int len, type;
```

The *gethostbyname* call returns the *hostent* data structure of the matching name. The name string can be obtained from call *gethostname*. The *gethostbyaddr* call returns the *hostent* data structure of the matching *addr*, *len* and *type*.

As we can see from the above description, the Internet address data structure is very complex. Fortunately, BSD UNIX has provided a group of functions for manipulating Internet addresses. They are described in *inet(3n)* of the BSD UNIX manual. We only describe the following call for our purpose:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

char *inet_ntoa(in)
struct in_addr in;
```

This call converts an Internet address to an ASCII string representing the address in "." notation (e.g. 139.130.118.102).

The following is a short program that prints out the host's information. It is named *phn.c* (print host name). By default it prints out the local host's information. If you specify a host name, it prints out the information for that host. The program follows:

```
/*
 * phn.c—print the host name and other information.
 * compile: cc -o phn phn.c
 * execution: phn
 *            phn hostname
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>  /* for inet_ntoa() */

#define MAXHOSTNAMELEN 32
main(argc, argv)
int argc;
char *argv[];
```

```
{
char hostName[MAXHOSTNAMELEN];
char *ptr;
struct hostent *hp;

  if (argc < 2) {
    if (gethostname(hostName, MAXHOSTNAMELEN) == -1)  /* get my host name */
      exit(1);
  } else
    strcpy(hostName, argv[1]);
  /* get host's info */
  if ((hp = gethostbyname(hostName)) == NULL) {
    fprintf(stderr, "phn: %s: unknow host\n", hostName);
    exit(1);
  }

  printf("Host name: %s.\n", hp->h_name);
  while ((ptr = *(hp->h_aliases)) != NULL) {
    printf("    Alias: %s\n", ptr);
    hp->h_aliases++;
  }
  printf("Host address type: %d, address length=%d.\n", hp->h_addrtype,
         hp->h_length);
  if (hp->h_addrtype == AF_INET)
    pr_inet(hp->h_addr_list, hp->h_length);
  else
    printf(stderr, "phn: Unknown address type");
}

pr_inet(listptr, length)
char **listptr;
int length;
{
  struct in_addr *ptr;
  while ((ptr = (struct in_addr *) *listptr++) != NULL)
    printf("Internet address: %s\n", inet_ntoa(*ptr));
}
```

If you simply execute the program by typing in *phn*, then the information for the local host is printed out. One possible output could be:

```
Host name: sky3.cm.deakin.edu.au.
    Alias: sky3
Host address type: 2, address length=4.
Internet address: 139.130.118.5
```

If you execute the program by specifying a host name, such as *phn turin*, then the information of host *turin* will be printed out as:

```
Host name: turin.cm.deakin.edu.au.
Host address type: 2, address length=4.
Internet address: 128.184.82.150
```

9.6.2. Socket Creation

Several system calls are provided to create, use and manage sockets. Before using a connection-oriented socket, one must first create it, bind it to a name and connect it to another socket. For a server, after creation and binding, the socket may listen for a connection and accept it. All these are performed by a group of system calls. In most cases, three files will be included before using these calls, they are *sys/types.h*, *sys/socket.h* and *netinet/in.h.*

To create a socket, use the socket system call:

```
descriptor = socket(domain, type, protocol);
int descriptor;
int domain;
int type;
int protocol;
```

The *domain* is, of course, AF_INET. The *type* can be currently SOCK_STREAM, SOCK_DGRAM, or SOCK_RAW, whereas the *protocol* specifies a particular protocol to be used by the socket. If a 0 (zero) is specified in the protocol parameter, the system will select the proper protocol for you.

Normally only a single protocol exists to support a particular socket type using a given address format (domain). The most used protocols in the Internet domain are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), and they are used for sockets of type SOCK_STREAM and SOCK_DGRAM, respectively.

The returned value is actually a small integer which represents a socket, and the user may use it in the later system calls which operate on sockets.

9.6.3. Name Binding

After creation, generally a name will be bound to the socket so that the user can then use that socket. The bind call is used to assign a name to an unnamed socket:

```
bind(descriptor, name, namelen);
int descriptor;
struct sockaddr name;          / for Internet domain */
int namelen;
```

Internet domain socket naming is more complex than that of UNIX domain, but if you know the host name and the communication port (both are local), then the following fragment would be used (suppose the host name is *sky3*, the port is 5100 and we also want to bind the name to *descriptor*:

```
#include <netinet/in.h>
#include <netdb.h>
#define machine “sky3”
struct sockaddr_in my_sock;
struct hostent *hp;
hp = gethostbyname(machine);              /* get host’s info */
bzero((char )&my_sock, sizeof(my_sock));/ clear my_sock to 0 */
my_sock.sin_family = hp->h_addrtype;   /* AF_INET domain */
my_sock.sin_port = htons(5100);                      /* set port */
bcopy(hp->h_addr, (char *)&my_sock. sin_addr, hp->h_length);
```

```
                /*get address */
if (bind(descriptor, (struct sockaddr *)&my_sock, sizeof(my_sock)) == -1)
                { /* error */}
```

9.6.4. Connection Establishment

In the Internet domain the following three system calls are used for socket connection: *listen, accept*, and *connect*.

Suppose we have socket *descriptorSer* in a server program and socket *descriptorCli* in a client program. If the server now is willing to offer its service, it uses a *listen* system call and then uses an *accept* system call to passively waits for the client to make connection. On the other hand, the *connect* system call is used by the client to initialise a connection. Please note, however, it is not necessary for UPD (connectionless) sockets to perform these steps.

The *listen* system call is quite simple:

```
listen(descriptorSer, 5);
```

Where the last parameter is the maximum number of outstanding connections which may be queued waiting to be accepted by the server to accept, and 5 is the system limitation of maximum connection on any one queue.

After listening, the server uses the *accept* call to accept a connection:

```
int cliLen; struct sockaddr in sockCli;    /* client socket address */ cliLen =
sizeof(sockCli);   descriptorCom   =   accept(descriptorSer,   (struct   sockarrd
*)&sockCli, &cliLen);
```

The returned value *descriptorCom* is a new socket and it is used in the input/output calls when needed. If the server wishes to find whom the client is, then several system calls can be applied to that socket, and they will return the client name *sockCli*.

The *connect* system call for a socket in the Internet domain looks like:

```
struct sockaddr in sockSer; connect(descriptorCli, (struct sockaddr *)&sockSer,
sizeof(sockSer));
```

In some situations the *connect* call will fail. These situations are very important when considering fault tolerance of the application system.

9.6.5. Transfer Data and Discard Sockets

Several system calls can be used to transfer data between connected sockets. The simplest group of such calls are *write* and *read* calls. They are identical to *write* and *read* for disk files:

```
char buf[BUFSIZE];
int msglen;
write(descriptor, buf, msglen);    /* send to correspondent */
read(descriptor, buf, sizeof(buf));     /* read from correspondent */
```

Where *descriptor* is the socket created by the client in a client program and is the value returned from the server's *accept* call in a server program.

In a *write* call, the array *buf* contains the message to be sent and *msglen* gives the number of bytes to be sent. If a character string is to be sent, *strlen(buf)* can be used for the message length. By default, *write* does asynchronous writes. That is, after the data is

written to a buffer cache, control returns to the program. The actual write to a device takes place after control returns. Upon successful completion, the number of bytes actually written is returned. Otherwise, a -1 is returned, and *errno* is set to indicate the error. The *errno* is the UNIX error number variable and is defined in *errno.h*. To use *errno*, the *errno.h* file must be included.

Upon successful completion, *read* returns the number of bytes actually read and placed in the array *buf*. The system returns the number of bytes requested (*sizeof(buf)*) if the descriptor references a stream which has that many bytes left before the *end-of-file*. If the returned value is 0, then *end-of-file* has been reached. Otherwise, a -1 is returned and the global variable *errno* is set to indicate the error.

Alternatively, programs might use the *send* and *recv* system calls as follows:

```
int flags;
send(descriptor, buf, sizeof(buf), flags);  /* send to correspondent */
recv(descriptor, buf, sizeof(buf), flags);  /* receive from correspondent */
```

Where *descriptor* in both server and client programs are the same as above. The *flags* can be 0 or can be specified explicitly. The most interesting flags are:

```
#define MSG_PEEK  0x1     /* look at data without reading */
#define MSG_OOB       0x2     /* process out-of-bound data */
```

A program can use the MSG_PEEK flag to look at the available data, without having the system discard the data after the *recv* call. The MSG_OOB flag specifies that the data to be sent or received is of type *out-of-bound* or *expedited*. With this type of data, we want the sending and receiving services to process these data before any other data that have been buffered. Out-of-bound type is only defined for stream sockets, and the UNIX domain stream sockets do not support it.

It is also possible to use the standard *stdio* to read the socket. In that case, a *fdopen* call can be used to open the socket for reading and the *fgetc* call can be used to read characters from the opened socked in the same manner as in file reading.

The above calls are generally used in connection-oriented communications. There are other data transmission calls that can be used at both connection-oriented and connectionless communication services. We mention the *sendto/recvfrom* pair here:

```
cc = sendto(descriptor, msg, len, flags, to, tolen)
int cc, descriptor;
char *msg;
int len, flags;
struct sockaddr_un *to;
int tolen;

cc = recvfrom(s, buf, len, flags, from, fromlen)
int cc, s;
char *buf;
int len, flags;
struct sockaddr *from;
int *fromlen;
```

In a *sendto* call, the address of the target is given by *to*, with *tolen* specifying the address size. The length of the message is given by *len*. If the message is too long to pass atomically through the underlying protocol, the error EMSGSIZE is returned, and the message is not transmitted.

The *sendto* call returns the number of characters sent, or -1 if an error occurred. Return values of -1 only indicate some locally detected errors. If message space is unavailable at the socket to hold the message to be transmitted, *sendto* blocks, unless the socket has been placed in *nonblocking* I/O mode.

In a *recvfrom* call, if *from* is nonzero, the source address of the message is filled in on return. The fromlen is a value-result parameter, initialised to the size of the buffer associated with *from*, and modified on return to indicate the actual size of the address stored there. The length of the message is returned in *cc*. If the message is too long to fit in the supplied buffer, excess bytes can be discarded, depending on the type of sending socket the message. That is, if the sending socket is a sequenced packet stream socket, the excess bytes will be discarded.

If no messages are available at the socket, the *recvfrom* call waits for a message to arrive, unless the socket is *nonblocking*. If the socket is *nonblocking* then a *cc* of -1 is returned, and the global variable *errno* is set to EWOULDBLOCK.

Once a socket *descriptor* is no longer of use, it may be discarded by applying a *close* system call to the socket:

```
close(descriptor);
```

Upon successful completion, a value of 0 is returned. Otherwise, a value of -1 is returned, and the global variable, *errno*, is set.

If there are some data associated with a SOCK_STREAM type socket when the *close* call takes place, the system will continue to attempt to transfer the data. However, if after about 4 minutes the data is still not delivered, it will be discarded. The *shutdown* system call can be used prior to a *close* call to discard the pending data if the user is not interested in it:

```
shutdown(descriptor, how);
int how;
```

Where *how* is 0 if the user is no longer interested in reading data, 1 if no more data will be sent, or 2 if no data is to be sent or received. Applying *shutdown* to a socket causes any data queued to be immediately discarded. A zero (0) is returned if the *shutdown* call succeeds, -1 if it fails.

9.7. Examples

9.7.1. Using Stream Sockets: A Simple Example

In this example, two programs use connection-oriented Internet sockets to communicate with each other. The server program is supposed to be executed first. It creates an Internet stream socket and binds the socket to a name, then it listens to the socket and waits for a connection. If a connection request arrives, the server reads in a message from the client,

displays it and writes back an acknowledgement to the client program. After that, the server program closes the socket and exits.

The server program is named *ismpser.c* indicating that it uses Internet domain sockets, and is a simple example server program.

By default, it uses stream sockets. The listing follows.

```
/* Simple example for Internet domain communications (stream socket)
   Server program (sequential)
   Compile: cc -o ismpser ismpser.c
   Execute: ismpser
   Note:    ismpser must be executed before ismpcli
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define MAXHOSTNAMELEN 32
#define SERVER_STREAM_PORT 5100
#define oops(msg) { perror(msg);  exit(-1); }

main()
{
int sockSer, sockPeer, cliLen;
int i;
struct sockaddr_in ser, cli;
char buf[BUFSIZ];
char myhostname[MAXHOSTNAMELEN];
struct hostent *hp;

  /* get server host's info */
  if (gethostname(myhostname, MAXHOSTNAMELEN) == -1)  /* get my host name */
    oops("gethostname");
  if ((hp = gethostbyname(myhostname)) == NULL)
    oops("gethostbyname");   /* get host's network address */
  /* create an Internet domain, stream socket */ if ((sockSer = socket(AF_INET,
SOCK_STREAM, 0)) == -1)
    oops("socket");
  /* bind the server socket to an name */
  bzero((char ) &ser, sizeof(ser));   / empty name buffer */
  ser.sin_family = AF_INET;                              /* Internet domain */
  ser.sin_port = htons(SERVER_STREAM_PORT);              /* port */
  bcopy(hp->h_addr, (char )&ser.sin_addr, hp->h_length); / host */
  if (bind(sockSer, &ser, sizeof(ser)) == -1)
    oops("bind");
  /* listen for connections */
  listen(sockSer, 5);
  /* accept connection from client */
  cliLen = sizeof(cli);
  if ((sockPeer = accept(sockSer, &cli, &cliLen)) == -1)
```

```
    oops("accept");
  /* accept data from the client */
  for (i=0; i<BUFSIZ; i++)  /* clean buf */
    buf[i] = '\0';
  if ((i = read(sockPeer, buf, sizeof(buf))) == -1)
    oops("read");
  printf("Server: Data received: %s. Length=%d.\n", buf, i);
  /* send data to the connected client */ strcpy(buf, "This is the reply message
from server program"); if (write(sockPeer, buf, strlen(buf)) == -1)
    oops("write");
  printf("Server: Data sent: %s\n", buf);
  /* close the two sockets completely */ if (close(sockSer) == -1 ||
close(sockPeer) == -1)
    oops("close");
  printf("Server: successful.\n");
}
```

The client program is supposes to be executed after the server program's execution. After the creation of an Internet stream socket, the client program initialises a connection to the server. If the connection succeeds, the client writes a message to the server and reads a reply from the server. The reply is then displayed and the client program exits.

The client program is named *ismpcli.c* indicating that it uses Internet domain sockets and is a simple example, client program.

By default, it uses stream sockets. The listing follows.

```
/* Simple example for Internet domain communications (stream socket)
   Client program
   Compile: cc -o ismpcli ismpcli.c
   Execute: ismpcli
   Note:    ismpser must be executed before ismpcli
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_STREAM_PORT 5100
#define SERVERHOSTNAME "berry.fcit.monash.edu.au"
#define oops(msg) { perror(msg);  exit(-1); }

main()
{
int sockCli;
int i;
struct sockaddr_in ser;
char buf[BUFSIZ];
struct hostent *hp;

  /* get server host's network address */
  if ((hp=gethostbyname(SERVERHOSTNAME))==NULL)
    oops("gethostbyname");
```

```
  /* create an Internet domain, stream socket */ if ((sockCli = socket(AF_INET,
SOCK_STREAM, 0)) == -1)
    oops("socket");
  /* fill in the server address buffer */
  bzero((char ) &ser, sizeof(ser));   / empty name buffer */
  ser.sin_family = AF_INET;                              /* Internet domain */
  ser.sin_port = htons(SERVER_STREAM_PORT);              /* port */
  bcopy(hp->h_addr, (char )&ser.sin_addr, hp->h_length); / host */
  /* connect to server, its name is in ser */
  if (connect(sockCli, &ser, sizeof(ser)) == -1)
    oops("connect");
  /* send a string to server */ strcpy(buf, "This is a request message from
client"); if (write(sockCli, buf, strlen(buf)) == -1)
    oops("write");
  printf("Client: Data sent: %s\n", buf);
  /* read in data from the server */
  if ((i = read(sockCli, buf, sizeof(buf))) == -1)
    oops("read");
  printf("Client: Data received: %s. Length=%d.\n", buf, i);
  /* close the socket */
  if (close(sockCli) == -1)
    oops("close");
  printf("Client: successful.\n");
}
```

The communication can only be executed once by using the above two programs. In order to execute the communication many times, either a looping server or a concurrent server program would be used. We present a concurrent version of the server program here.

The concurrent version of the server program is named *ismpsercon.c* indicating that it uses Internet domain sockets and is a simple example, server program, concurrent version.

By default, it uses stream sockets. The listing follows.

```
/* Simple example for Internet domain communications (stream socket)
   Server program (concurrent)
   Compile: cc -o ismpser ismpser.c
   Execute: ismpser
   Note:    ismpser must be executed before ismpcli
*/

#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define MAXHOSTNAMELEN 32
#define SERVER_STREAM_PORT 5100
#define oops(msg) { perror(msg);  exit(-1); }

main()
```

```
{
int sockSer, sockPeer, cliLen;
struct sockaddr_in ser, cli;
char buf[BUFSIZ];
int mainPid, kidPid;
char myhostname[MAXHOSTNAMELEN];
struct hostent *hp;

  /* get server host's info */
  if (gethostname(myhostname, MAXHOSTNAMELEN) == -1)  /* get my host name */
    oops("gethostname");
  if ((hp=gethostbyname(myhostname))==NULL)  /* get host's network address */
    oops("gethostbyname");
  /* create an Internet domain, stream socket */ if ((sockSer = socket(AF_INET,
SOCK_STREAM, 0)) == -1)
    oops("socket");
  /* bind the server socket to an name */
  bzero((char ) &ser, sizeof(ser));   / empty name buffer */
  ser.sin_family = AF_INET;                              /* Internet domain */
  ser.sin_port = htons(SERVER_STREAM_PORT);              /* port */
  bcopy(hp->h_addr, (char )&ser.sin_addr, hp->h_length); / host */
  if (bind(sockSer, &ser, sizeof(ser)) == -1)
    oops("bind");
  /* listen for connections */
  listen(sockSer, 5);

  while (1) {
    /* accept connection from client */
    cliLen = sizeof(cli);
    if ((sockPeer = accept(sockSer, &cli, &cliLen)) == -1)
      oops("accept");
    mainPid = getpid();  /* parent pid */
    /* communication processing */
    kidPid = commProc(sockPeer, mainPid, sockSer);

    /* parent process, ready to accept new connections */
    close(sockPeer);
  }
}

/*
 * Communication processing. Accept from client then reply.
 * If shutdown received, kill both child and parent processes.
*/
int commProc(sockDesc, mpid, sockSer)
int sockDesc;  /* peer socket descriptor */
int mpid;      /* parent's process pid */
int sockSer;   /* server socket descriptor */
{
  char buf[BUFSIZ];
  int i, kidPid;

  /* fork to two processes */
  if ((kidPid = fork()) != 0)
    return (kidPid);  /* return to parent */
```

```
  /* child process. Accept data from the client */
  for (i=0; i<BUFSIZ; i++)  /* clean buf */
    buf[i] = '\0';
  if ((i = read(sockDesc, buf, sizeof(buf))) == -1)
    oops("read");
  printf("Server: Data received: %s. Length=%d.\n", buf, i);
  if (strcmp(buf, "shutdown") == 0) { /* shutdown request */
    /* kill parent process */
    if (kill(mpid, SIGINT) == -1)
      perror("kill");

    /* close the two sockets completely */
    if (close(sockSer) == -1 || close(sockDesc) == -1)
      oops("close");
    /* kill child process */
    if (kill(kidPid, SIGINT) == -1)
      perror("kill");
    exit(0);
  }

  /* come here when it is a normal request message */
  /* send data to the connected client */
  strcpy(buf, "This is the reply message from server program");
  if (write(sockDesc, buf, strlen(buf)) == -1)
    oops("write");
  printf("Server: Data sent: %s\n", buf);
  exit(0);
}
```

The "shutdown" message can be sent by modifying the client program.

9.7.2. Using Datagram Sockets: A Simple Example

Next is an example of using Internet datagram sockets for simple communications. The example performs the same functions as the previous one. The main differences between using a datagram socket and using a stream socket are the socket creation and the data transfer calls. In this example program, we create datagram sockets and use *sendto/recvfrom* calls to transfer data between server and client programs.

The server program is named ismdser.c indicating that it uses Internet sockets and is a simple example, using datagram sockets , server program. The listing follows.

```
/* Simple example for Internet domain communications (datagram socket)
   Server program (sequential)
   Compile: cc -o ismdser ismdser.c
   Execute: ismdser
   Note:    ismdser must be executed before ismdcli
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
```

```
#define MAXHOSTNAMELEN 32
#define SERVER_DGRAM_PORT 5200
#define oops(msg) { perror(msg);  exit(-1); }

main()
{
int sockSer, cliLen;
int i;
struct sockaddr_in ser, cli;
char buf[BUFSIZ];
char myhostname[MAXHOSTNAMELEN], forhostname[MAXHOSTNAMELEN];
struct hostent *hp;
  /* get server host's info */
  if (gethostname(myhostname, MAXHOSTNAMELEN) == -1)  /* get my host name */
    oops("gethostname");
  if ((hp = gethostbyname(myhostname)) == NULL)
    oops("gethostbyname");   /* get host's network address */
  /*  create  an  Internet  domain,  datagram  socket  */  if  ((sockSer  =
socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    oops("socket");
  /* bind the server socket to an name */
  bzero((char ) &ser, sizeof(ser));   / empty name buffer */
  ser.sin_family = AF_INET;                              /* Internet domain */
  ser.sin_port = htons(SERVER_DGRAM_PORT);             /* port */
  bcopy(hp->h_addr, (char )&ser.sin_addr, hp->h_length); / host */
  if (bind(sockSer, &ser, sizeof(ser)) == -1)
    oops("bind");
  /* accept data from the client */
  for (i=0; i<BUFSIZ; i++)  /* clean buf */
    buf[i] = '\0';
  bzero((char ) &cli, sizeof(cli));   / empty name buffer */
  cliLen = sizeof(cli);
  if ((i = recvfrom(sockSer, buf, sizeof(buf), 0, &cli, &cliLen)) == -1)
    oops("recvfrom");
  /* get the client's host name and port */
  if ((hp = gethostbyaddr((char *)&cli.sin_addr, sizeof(struct in_addr),
                 AF_INET)) == NULL)
    oops("gethostbyaddr");
  printf("Server: Request from: host: %s, port: %d.\n",
                 hp->h_name, ntohs(cli.sin_port));
  printf("Server: Received data: %s. Length=%d.\n", buf, i);
  /* send data to the corresponding client */ strcpy(buf, "This is the reply
message from server program"); if (sendto(sockSer, buf, strlen(buf), 0, &cli,
sizeof(cli)) == -1)
    oops("sendto");
  printf("Server: Data sent: %s\n", buf);
  /* close the socket completely */
  if (close(sockSer) == -1)
    oops("close");
  printf("Server: successful.\n");
}
```

The client program is named *ismdcli.c* indicating that it uses Internet domain sockets and is a simple example, using datagram sockets, client program. Its listing follows.

```
/* Simple example for Internet domain communications (datagram socket)
   Client program
   Compile: cc -o ismdcli ismdcli.c
   Execute: ismdcli
   Note:    ismdser must be executed before ismdcli
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define MAXHOSTNAMELEN 32
#define SERVER_DGRAM_PORT 5200
#define CLIENT_DGRAM_PORT 5201
#define SERVERHOSTNAME “berry.fcit.monash.edu.au”
#define oops(msg) { perror(msg);  exit(-1); }

main()
{
int sockCli;
int i;
struct sockaddr_in ser, cli;
char buf[BUFSIZ];
char myhostname[MAXHOSTNAMELEN];
struct hostent *hp;

  /* get client host’s info */
  if (gethostname(myhostname, MAXHOSTNAMELEN) == -1)  /* get my host name */
    oops(“gethostname”);
  if ((hp = gethostbyname(myhostname)) == NULL)
    oops(“gethostbyname”);   /* get host’s network address */
  /*  create  an  Internet  domain  datagram  socket  */  if  ((sockCli  =
socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    oops(“socket”);
  /* bind the client socket to an name */
  bzero((char ) &cli, sizeof(cli));   / empty name buffer */
  cli.sin_family = AF_INET;                              /* Internet domain */
  cli.sin_port = htons(CLIENT_DGRAM_PORT);             /* port */
  bcopy(hp->h_addr, (char )&cli.sin_addr, hp->h_length); / host */
  if (bind(sockCli, &cli, sizeof(cli)) == -1)
    oops(“bind”);
  /* get server host’s network address */
  if ((hp=gethostbyname(SERVERHOSTNAME))==NULL)
    oops(“gethostbyname”);
  /* fill in the server address buffer */
  bzero((char ) &ser, sizeof(ser));   / empty name buffer */
  ser.sin_family = AF_INET;                              /* Internet domain */
  ser.sin_port = htons(SERVER_DGRAM_PORT);             /* port */
  bcopy(hp->h_addr, (char )&ser.sin_addr, hp->h_length); / host */
```

```
  /* send a string to server */ strcpy(buf, "This is a request message from
client"); if (sendto(sockCli, buf, strlen(buf), 0, &ser, sizeof(ser)) == -1)
    oops("sendto");
  printf("Client: Data sent: %s\n", buf);
  /* read in data from the server */ if ((i = recvfrom(sockCli, buf,
sizeof(buf), 0, (char *)0, (int *)0)) == -1)
    oops("recvfrom");
  buf[i] = '\0';
  printf("Client: Received: %s. Length=%d.\n", buf, i);
  /* close the socket */
  if (close(sockCli) == -1)
    oops("close");
  printf("Client: successful.\n");
}
```

Similarly we have a concurrent version of the Internet datagram socket example. The concurrent server program is named *ismdsercon.c* indicating that it uses Internet domain sockets and is a simple example, using datagram sockets, server program, concurrent version. Its listing follows.

```
/* Simple example for Internet domain communications (datagram socket)
   Server program (concurrent)
   Compile: cc -o ismdsercon ismdsercon.c
   Execute: ismdsercon
   Note:    ismdsercon must be executed before ismdcli
*/

#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define MAXHOSTNAMELEN 32
#define SERVER_DGRAM_PORT 5200
#define oops(msg) { perror(msg);  exit(-1); }

main()
{
int sockSer, cliLen, i;
struct sockaddr_in ser, cli;
char buf[BUFSIZ];
int mainPid, kidPid;
char myhostname[MAXHOSTNAMELEN], forhostname[MAXHOSTNAMELEN];
struct hostent *hp;
  /* get server host's info */
  if (gethostname(myhostname, MAXHOSTNAMELEN) == -1)  /* get my host name */
    oops("gethostname");
  if ((hp=gethostbyname(myhostname))==NULL)  /* get host's network address */
    oops("gethostbyname");
  /* create an Internet domain datagram socket */ if ((sockSer =
socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    oops("socket");
  /* bind the server socket to an name */
```

```
  bzero((char ) &ser, sizeof(ser));   / empty name buffer */
  ser.sin_family = AF_INET;                              /* Internet domain */
  ser.sin_port = htons(SERVER_DGRAM_PORT);             /* port */
  bcopy(hp->h_addr, (char )&ser.sin_addr, hp->h_length); / host */
  if (bind(sockSer, &ser, sizeof(ser)) == -1)
    oops("bind");
  while (1) {
    /* accept connection from client */
    cliLen = sizeof(cli);
    if ((i = recvfrom(sockSer, buf, sizeof(buf), 0, &cli, &cliLen)) == -1)
      oops("recvfrom");
    buf[i] = '\0';

    mainPid = getpid();  /* parent pid */
    /* communication processing */
    kidPid = commProc(sockSer, mainPid, buf, cli);

    /* parent process, ready to accept new connections */
  }
}

/*
 * Communication processing. Accept from client then reply.
 * If shutdown received, kill both child and parent processes.
*/
int commProc(sockDesc, mpid, bf, cli)
int sockDesc;  /* socket descriptor */
int mpid;      /* parent's process pid */
char *bf;
struct sockaddr_in cli;
{
  char buf[BUFSIZ];
  int kidPid;
  struct hostent *hp;

  /* fork to two processes */
  if ((kidPid = fork()) != 0)
    return (kidPid);  /* return to parent */
  /* child process */
  /* get the client's host name and port */
  if ((hp = gethostbyaddr((char *)&cli.sin_addr, sizeof(struct in_addr),
                  AF_INET)) == NULL)
    oops("gethostbyaddr");
  printf("Server: Request from: host: %s, port: %d.\n",
                  hp->h_name, ntohs(cli.sin_port));
  printf("Server: Received data: %s\n", bf);
  if (strcmp(bf, "shutdown") == 0) { /* shutdown request */
    /* kill parent process */
    if (kill(mpid, SIGINT) == -1)
      perror("kill");

    /* close the two sockets completely */
    if (close(sockDesc) == -1)
      oops("close");
```

```
    /* kill child process */
    if (kill(kidPid, SIGINT) == -1)
      perror("kill");
    exit(0);
  }

  /* come here when it is a normal request message */
  /* send data to the connected client */
  strcpy(buf, "This is the reply message from server program");
  if (sendto(sockDesc, buf, strlen(buf), 0, &cli, sizeof(cli)) == -1)
    oops("sendto");
  printf("Server: Data sent: %s\n", buf);
  exit(0);
}
```

10. An Application Example: Electronic Mails

10.1. Study Points

- Understand the basic concepts of electronic mail systems.
- Understand the architecture of an email system.
- Be familiar with the basic features of various protocols related to the email service.
- Understand the transaction methods on the Internet.
- Identify the various types of electronic paymehts

References: RFC 822 and RFC 821.

10.2. Mailboxes and Addresses

Electronic mail (email) was originally designed as a straightforward extension of a traditional office memo. Like an office memo, an email message is created by one person, and copies are sent to one or more persons. It is designed, like an office memo, to be convenient and to require less overhead than more formal communication. Unlike a traditional memo, an email is automated and can be processed by programs. Therefore, an application program can send emails to users and copies of an email message can be propagated to many users quickly.

An email system functions a similar way as a traditional office environment in which each person has a mailbox; whenever a memo is created, the creator designates one or more recipients; a clerk then places a copy of the memo in each recipient's mailbox. An electronic mailbox consists of a passive storage area such as a file on a disk. The mailbox is private, in which only the owner can examine or remove messages.

Each electronic mailbox has a unique email address. Each email address specifies both the mailbox and a computer, in a format such as mailbox@computer, where the computer is a string that denotes the computer on which the mailbox is located (a domain name).

10.3. Message Format

The main standards that related to the protocols of email transmission and reception are:

- Simple Mail Transfer Protocol (SMTP) that uses the TCP/IP protocol suite. It has been traditionally limited to the text-based email systems.
- Multipurpose Internet Mail Exchange (MIME) that allows the transmission and reception of mails that contain multiple types of data, such as speech, images, and video. It is a newer standard than SMTP and uses much of its basic protocol.

An email message consists of two parts separated by a blank line. The first part is a header that contains information about the sender, recipients, and subject of the message. The second part is a body that contains the text of the message.

Each header line begins with a keyword followed by a colon and additional information. The keyword tells the email software how to interpret the reminder of the line. Some

keywords are required in each header and others are optional. For example, each header must contain a keyword *To* and specifies a list of recipients. The reminder of the line followed by *To* and a colon contains a list of one or more email addresses, where each email address is a recipient. Email software system places a line that begins with the keyword *From* followed by the email address of the sender in the header of each message. The *Date* and *Subject* keywords are optional. The following table lists some of the commonly used keywords:

Keyword	**Meaning**
From	Sender's address
To	Recipient's addresses
Cc	Addresses for carbon copies
Date	Date on which message was sent
Subject	Topic of the message
Reply-To	Address to which reply should go
X-Charset	Character set used (usually ASCII)
X-Mailer	Mail software used to send the message
X-Sender	Duplicate of sender's address
X-Face	Encoded image of the sender's face

If an email software does not understand a header line, it processes it through unchanged. Therefore, additional header lines can be added to suit various extensions to the basic email services

The MIME protocol allows a sender and receiver to choose an encoding for the binary files in messages. It also allows a sender to divide a message into several parts and to specify the encoding for each part independently. Thus, a user can send both a text message and an image in the same message.

MIME adds two lines to an email header: one to declare that MIME was used in creating the message, and another to specify how MIME information is included in the body. The chief advantage of MIME is its flexibility: the standard does not specify a single encoding scheme that all senders and receivers must use. Instead, MIME allows new encoding to be used at any time, provided both sender and receiver agree with the new scheme and a unique name for it. MIME is also compatible with the older email systems, i.e., the additional lines in MIME messages are transferred by the older systems without any changes and interpretation.

10.4. Architecture of an Email System

Figure 10.1 shows a typical email system architecture.

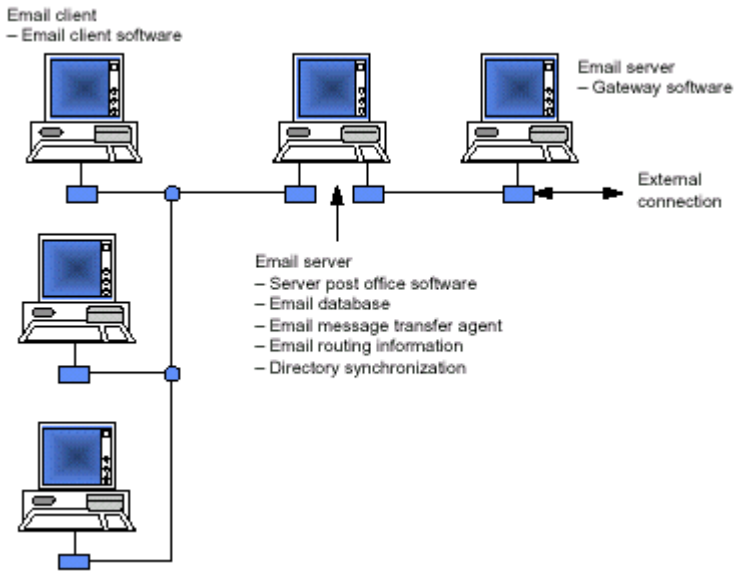


Figure 10.1. Architecture of an email system

A *post office* buffers on-going email messages before they are transmitted and stores the incoming messages. It also runs the server software capable of routing messages (a message transfer agent) and maintaining the post office database. The *message transfer agent* is responsible of forwarding messages between post offices and destination email clients. This software can either resides on the local post office or on a physically separate server. *Gateways* provide part of the message transfer agent functionality. They translate between different email systems, different email protocols, and different address schemes. *Email clients* are normally the computers connected to the post office. They consist three parts:

- Email Application Program Interface (API), such as MAPI, VIM, MHS, and CMC.
    - MAPI (message API): Microsoft part of Windows Operating System Architecture.
    - VIM (Vendors-Independent-Messaging): Lotus, Novell, Apple, and Borland derived email API.
    - MHS (Message Handling Service): Novell network interface which is opten used as an email gateway protocol.
    - CMC (Comman Mail Call): Email API associated with the X.400 native messaging protocol.
- Message protocols, such as SMTP and X.400. The SMTP is defined in RFC822 and RFC821. The X.400 is an OSI-defined standard for email message delivery. It is becoming extinct because of its complexity and poor design.
- Network transport protocols, such as Ethernet, FDDI, etc.

An email system can be configured using a shared-file approach or client-server approach, as depicted in Figure 10.2.. In a shared-file system (also called store and forward), the source mail client sends the mail message to the local post office. This post

office then transfers control to a message transfer agent which then stores the message for a short while before sending it to the destination opst office. The destination mail client periodically checks its own post office to determine if it has mail for it.

In the client-server approach the source client sets up a real-time connection with the local post office, which then sets up a real-time connection with the destination, which in turn sets up a real-time remote connection with the destination client. The message will then arrive at the destination client when all these connections are established.

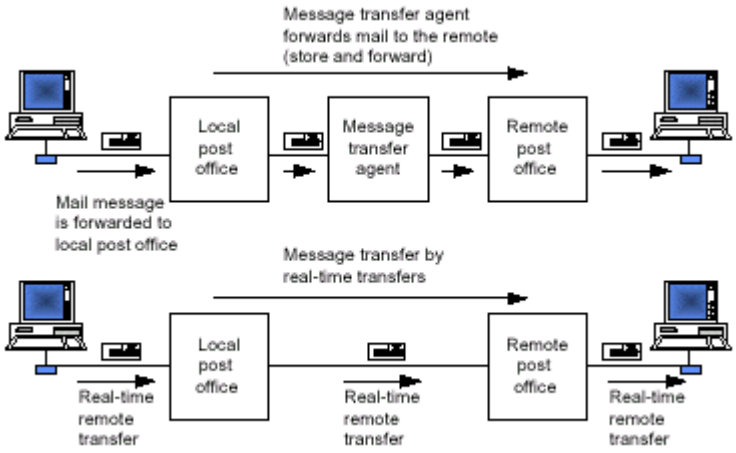


Figure 10.2. Email system configurations.

10.5. Mailbox Access

Mailboxes can only be placed on computers with mail servers, with constant Internet connection, and normally with large amount of storage. These computers are normally powerful since multiple users need to access the mail system simultaneously. Thus a personal computer is not usually used to run a mail server. Therefore, a protocol is needed to allow a user's mailbox to reside on a computer that runs a mail server, and allows the user to access items in the mailbox from other computers. This protocol is known as the *Post Office Protocol* (POP). It requires an additional server using the POP protocol to run on the computer with the mailbox. A user runs email software that becomes a client of the POP server to access the contents of the mailbox. Figure 10.3 illustrates this structure.

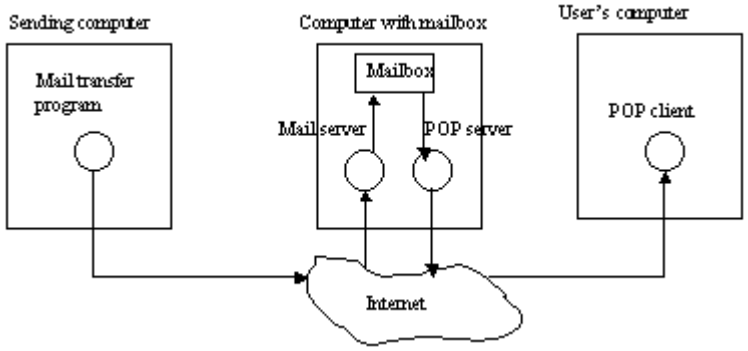


Figure 10.3. POP server and client.

The email server and the POP server have several differences. First, the mail server uses the SMTP protocol, while the POP server uses the POP protocol. Second, the mail server accepts a message from any sender, while the POP server only allows a user to access the mailbox after the authentication. Third, the mail server can transfer only email messages, while the POP server can provide information about the mailbox contents.

11. Network Security

11.1. Study Points

- Understand the basic concepts of a secure network.
- Understand the principles of a private key encryption and a public key encryption.
- Understand the concepts of digital signatures, packet filters, and firewalls.

11.2. Secure Networks

11.2.1. What is a Secure Network?

A work on any types of networking, including high-speed networks, is not complete without a discussion on network security. Networks cannot be simply classified as secure or not secure since the term of "secure" is not absolute: each group of users may define the level of security differently. For example, some organizations may regard the stored data as valuable and require that only authenticated users gaining access to these data. Some organizations allow outside users to browse their data, but prevent the data to be altered by outside users. Some organizations may regard the communication as the most important issue in network security and require that the messages be kept private and that the senders and recipients be authenticated. Yet many organizations need some combinations of the above requirements. Therefore, the first step for an organization to building a secure network is to define its security policy. The security policy specifies clearly and unambiguously the items that are to be protected.

A number of issues need to be considered in defining a security policy. They include the assessment of the values of information within an organization and the assessment of the costs and benefits of various security policies. Generally speaking, the following three aspects of security can be considered:

- Data integrity: it refers to the correctness of data and the protection from changes.
- Data availability: it refers to the protection against disruption of services.
- Data confidentiality and privacy: they refer to the protection against snooping or wiretapping.

11.2.2. Integrity Mechanisms and Access Control

The techniques used to ensure the integrity of data against accidental damage are the checksums and cyclic redundancy checks (CRC). To use such techniques, a sender compute a small, integer value as a function of the data in a packet. The receiver re-computes the function from the data that arrives, and compares the result to the value that the sender computed.

However, the checksums or the CRC cannot absolutely guarantee data integrity. For example, a planned attacker can alter the data and then can create a valid checksum for the altered data.

The password mechanism is used in most computer system to control access to resources. This method works well in a conventional computer system but may be vulnerable in a networked environment. If a user at one location sends a password across a network to a computer at another location, anyone who wiretaps the network can obtain a copy f the password. Wiretapping is easy when packets travel across a LAN because many LAN technologies permit an attached station to capture a copy of all traffic. In such a situation, additional steps must be taken to prevent passwords from being reused.

11.3. Data Encryption

11.3.1. Encryption Principles

Encryption is a method that transforms the information in such a way that it cannot be understood by anyone except the intended recipient who possesses a secret method that makes it possible to decrypt the message. Figure 11.1 depicts the encryption process.

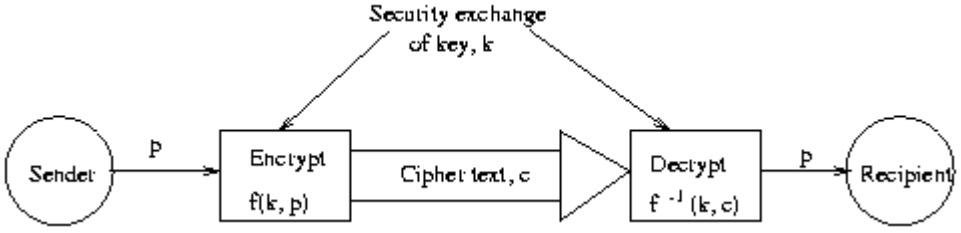


Figure 11.1. Single (private) key encryption

The two most popular private key techniques are DES (Data Encryption Standard) and IDEA (International Data Encryption Algorithm). The problem with the above scheme is the secured exchange of the key *k*: how can we be sure the key is known by both parties at the first place? A key distribution server sometimes is used to supply secret keys to clients. Figure 11.2 illustrates an example of key distribution server. Here user *A* needs to communicate with user *B*. A key is needed for both parties.

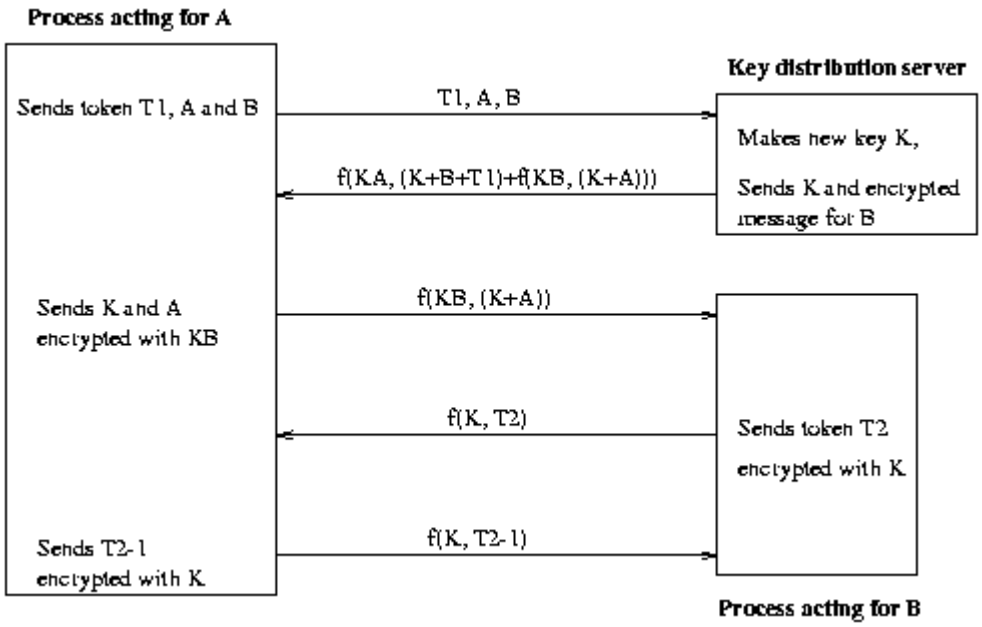


Figure 11.2. Key distribution server

A well known encryption method is the *public key encryption*. It uses two different keys (encryption key K_e and decryption key K_d), K_e is known to the sender and K_d the recipient. K_e can be made known publicly for use by anyone who wants to communicate, while K_d is kept secret. The RSA (after its inventors Rivest, Shamir, and Adleman) technique is one of the most popular public key encryption techniques and is based on the difficulty of factoring large numbers.

Here is an example: *A* (the Receiver) requires some secret information from *B* (the Sender). *A* generates a pair of keys K_e and K_d. K_d is kept secret and K_e is sent to *B*. *B* uses `c = E(Ke, p)` to encrypt the message and sends *c* to *A*. *A* then decrypts the message *c* using `p = D(Kd, c).` Figure 11.3 depicts this process.

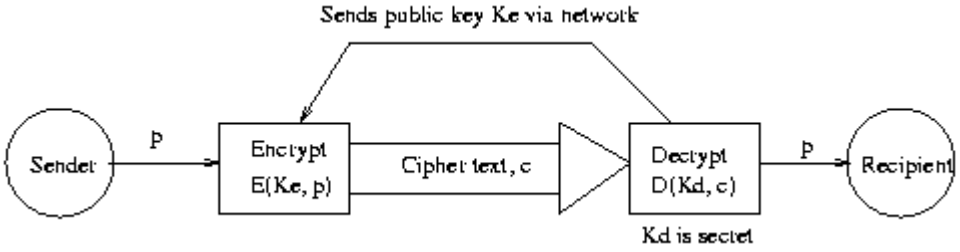


Figure 11.3. Public key encryption

11.3.2. Decryption

Many institutions and individuals read data that is not intend for them. They include:

- Government agencies. Traditionally governments around the world have reserved the rights to tape into any communication they think may be against the national interests.

- Spies who tap into communication for government and industry information.
- Individuals who like to read other people's messages.
- Individuals who hacks into systems and read sensitive information.
- Criminals who intercept information in order to use it for crime, such as intercepting PIN numbers on back accounts.

For example, the US government has proposed to beat encryption by trying to learn everyone's encryption key with the Clipper chip. The US government keeps a record of all the series numbers and encryption keys for each Clipper chip manufactured.

No matter how difficult an encryption is, every code is crackable and the measure of the security of a code is the amount of time it takes persons not addressed in the code to break the code. Normally to break a code, a computer tries all the possible keys until it finds the match. Thus a 1-bit code only has two keys. A 2-bit code would have 4 keys, and so on. For a 64-bit code it has 18,400,000,000,000,000,000 different keys. If one key is tested every 10μs, then it would take $1.84*10^{14}$ seconds (or 5,834,602 years).

However, as the improvement of computer power and techniques in parallel processing, the time used to crack a code may decrease dramatically. For example, if we think 1 million years would be safe for a code and we assume an increase of computer power of a factor of 2 every year, then it would take 500,000 years the next year. The same code would then be cracked in 1 year after 20 years. If we use parallel processing techniques then the code would be cracked much sooner.

11.4. Security Mechanisms on the Internet

11.4.1. Digital Signatures

Digital signatures are widely used on the Internet to authenticate the sender of a message. To sign a message, the sender encrypts the message using a key known only to the sender. The recipient uses the inverse function to decrypt the message. The receiver knows who sent the message because only the sender has the key needed to encrypt the message. A public key technique can be used in such a situation.

To sign a message, a sender encrypts the message using the private key. To verify the signature, the recipient looks up the user's public key and uses it to decrypt the message. Because only the user knows the private key, only the user can encrypt a message that can be decoded with the public key.

Interestingly, two levels of encryption can be used to guarantee that a message is both authentic and private. First, the message is signed by using the sender's private key to encrypt it. Second, the encrypted message is encrypted again using the recipient's public key. Here is the expression:

```
X = encrypt(public-rec, encrypt(private-sen, M))
```

Where *M* denotes a message to be sent, *X* denotes the string results from the two-level encryption, *private-sen* denotes the sender's private key, and *public-rec* denotes the recipient's public key.

At the recipient's side, the decryption process is the reverse of the encryption process. First, the recipient uses the private key to decrypt the message, resulting a digitally signed message. Then, the recipient uses the public key of the sender to decrypt the message again. The process can be expressed as follows:

```
M = decrypt(public-sen, decrypt(private-rec, X))
```

Where X is the encrypted message received by the recipient, M is the original message, private-rec denotes the recipient's private key, and public-sen denotes the sender's public key.

If a meaningful message results from the double decryption, it must be true that the message was confidential and authentic.

11.4.2. Packet Filtering

To prevent each computer on a network from accessing arbitrary computers or services, many sites use a technique known as *packet filtering*. A packet filter is a program that operates in a router. The filter consists of software that can prevent packets from passing through the router on a path from one network to another. A manager must configure the packet filter to specify which packets are permitted and which should be blocked. Figure 11.4 illustrates such a filter.

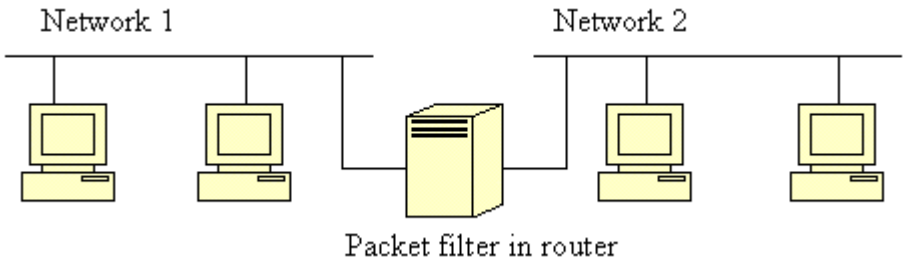


Figure 11.4. Packet filter in a router

A packet filter is configured to examine the packet header of each packet in order to decide which packets are allowed to pass through from one network to another. For example, the source and destination fields of a packet are examined to determine the if the packet is to be blocked or not. The filter can also examine the protocol of each packet or high-level services to block the access of a particular protocol or service For example, a packet filter can be configured to block all WWW access but to allow for email packets.

11.4.3. Internet Firewall

A packet filter can be used as a firewall for an organization to protect its computers from unwanted Internet traffic, as illustrated in Figure 11.5.

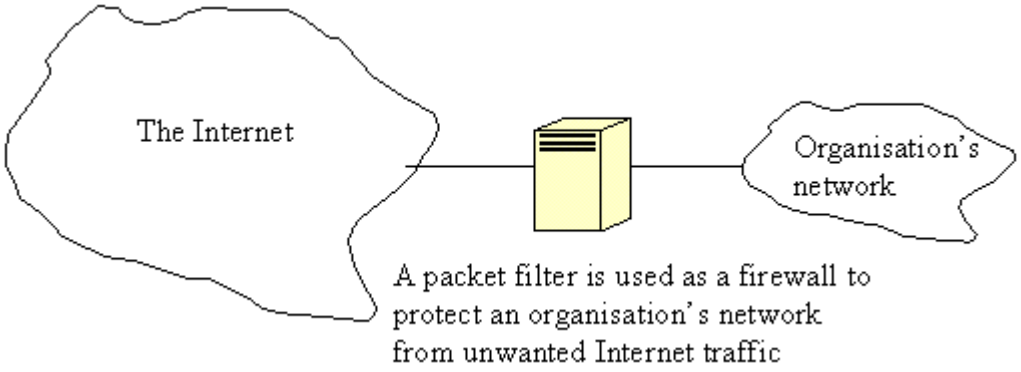


Figure 11.5. Internet firewall

Like a conventional firewall, an Internet firewall is designed to keep problems in the Internet from spreading into an organization's computer network. Without a firewall, an organization has to make all its computers secure to prevent unwanted Internet traffic. With a firewall, however, the organization can save money by only install the firewall and configure it to meet the requirements.

11.5. Review Questions

1. Use an examples to explain the processes of a private key encryption and a public key encryption.
2. Explain why the following claim is true: "If a meaningful message results from the double decryption, it must be true that the message was confidential and authentic".
3. What is a digital signature and how does it work?
4. What is a packet filter and how does it work?
5. What are the similarities and differences of a packet filter and an Internet firewall?