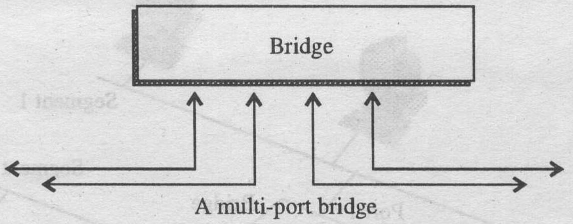
Also, bridges may consist of a number of interfaces or ports.



A multi-port bridge

Bridges are devices that pass packets between two LANs. The biggest benefit of a bridge is to create smaller networks, which can greatly increase the overall performance. In a LAN sharing a common wire, when one computer is talking all others must be listening. If a network of 100 computers is broken into two networks of 50 computers the overall bandwidth is doubled. The bridge connecting these two LAN segments allows a computer on one LAN to talk to a computer on the other LAN.

Like repeaters, bridges connect physically isolated networks to form a single logical network; however, a bridge has a little more intelligence and can provide some translation between dissimilar protocols.

For example, our token-passing segment wants to communicate with our CSMA/CD segment. The bridge will "repackage" the message from the token-passing segment into a format that the CSMA/CD segment will understand. Then, the bridge will act as a workstation on the CSMA/CD segment and contend for access. The same thing happens in reverse. A message is sent from the CSMA/CD segment to the token-passing segment. The bridge then "repackages" the message into a format the token-passing segment is expecting and waits for the token, just like any other workstation. An important point to remember is that a bridge will pass on any message it receives. Because the bridge is not smart enough to know that unlike LANs do not understand each other, it will go ahead and send the message. Because the two LANs speak a different "language", the message will be ignored.

13.5.1 How Bridges Work?

The most basic type of bridge attaches two or more LAN segments. The interface between a bridge and each LAN segment is known as a port. LANs attached to each port are called a network segment, as illustrated in the figure 13.7.

The bridge listens for every frame transmitted on each of its attached network segments, and compares each frame address to the bridge table maintained by its software. When a frame's destination is a device on a network segment other than the one on which it was transmitted, the bridge forwards the frame to the port connected to the destination segment. By forwarding only frames addressed to devices on other segments, bridges increase the effective throughput of the overall internetwork.

Bridges may be installed for any of the following reasons:

- To extend the distance or increase the number of devices that are part of the entire network.

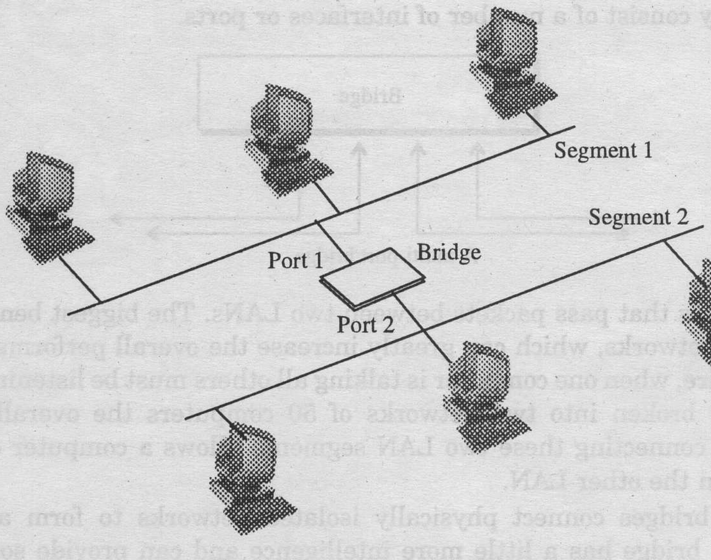


Fig. 13.7 Bridge ports and segments

- To improve the performance of the individual networks by reducing traffic. An excessive number of devices on a single network may affect network performance. It can be beneficial to split the network and connect the segments with a bridge.
- To link unlike networks such as Ethernet and Token-ring and forward packets between them.

13.5.2 Functions of Bridges

The various functions of bridges are discussed below:

1. **Frame forwarding and filtering :** When a data packet arrives at the bridge, the bridge reads the destination address and decides if the packet should be forwarded across the bridge or filtered and not sent. Without this function, all packets would be sent to all network segments. The purpose is to prevent locally addressed packets from crossing the bridge. A locally addressed packet is one where the network portion of the destination address is the same as the network portion of the source address-the packet is going to the same network from where it was sent.
2. **Loop resolution :** In an environment with many installed bridges, looping may occur. This is when a packet travels continuously over the network causing excess and unnecessary network traffic. Looping may be prevented or at least reduced by proper network design. Also, some bridges are able to detect looping packets and intercept them.
3. **Address tables :** Bridges need to know which packets to pass over the bridge and which packets to filter. Initially, with early bridges, address tables showing the locations of all attached devices had to be entered and updated by hand. Today, bridges are able to learn network addresses and build the address tables themselves.

They do so by examining packet flow-Transparent bridging-or by obtaining information through the use of explorer packets-Source Route bridging.

13.5.3 Classification of Bridges

Bridges can be broadly categorized into two groups-

- Local bridges
- Remote bridges

Local bridges provide a direct connection between multiple LAN segments in the same area. Remote bridges connect multiple LAN segments in different areas, usually over telecommunications lines. They provide connections over extended distances and have ports for connections to analog or digital telecommunications links. Remote bridges make use of MAN or WAN connectivity.

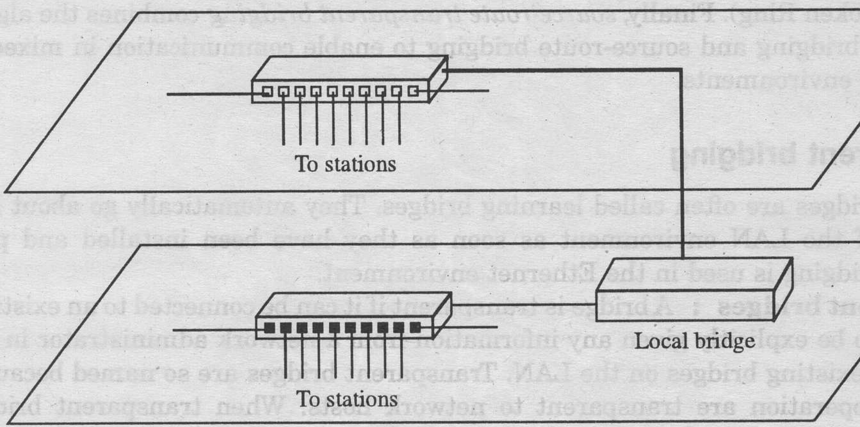


Fig. 13.8 Local bridging

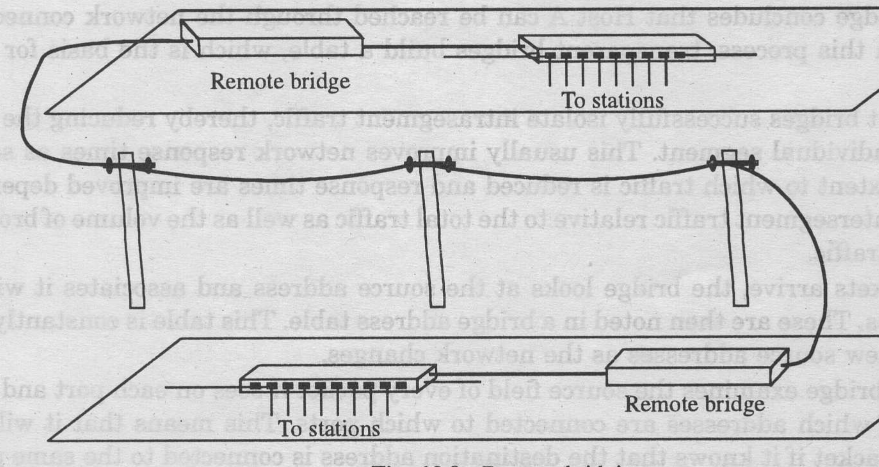


Fig. 13.9 Remote bridging

The critical part of a bridge is the software that builds and maintains its bridge table. Without this software, network administrators would have to manually create and maintain these tables. A bridge is categorized by the method or algorithm, that its software uses to build a bridge table. There are several types of bridging algorithms commonly used and each type has its own advantages and disadvantages. The basic types of bridging algorithms are stated below:

1. Transparent Bridging.
2. Spanning tree Bridging.
3. Source-routing Bridging.
4. Source-routing transparent Bridging.

Transparent bridging is found primarily in Ethernet environments, while *source-route bridging* occurs primarily in Token Ring environments. *Translational bridging* provides translation between the formats and transit principles of different media types (usually Ethernet and Token Ring). Finally, *source-route transparent bridging* combines the algorithms of transparent bridging and source-route bridging to enable communication in mixed Ethernet/Token Ring environments.

1. Transparent bridging

Transparent bridges are often called learning bridges. They automatically go about learning the topology of the LAN environment as soon as they have been installed and powered. Transparent bridging is used in the Ethernet environment.

Transparent bridges : A bridge is transparent if it can be connected to an existing LAN and not need to be explicitly given any information from a network administrator in order to work with the existing bridges on the LAN. Transparent bridges are so named because their presence and operation are transparent to network hosts. When transparent bridges are powered on, they learn the network's topology by analyzing the source address of incoming frames from all attached networks. If, for example, a bridge sees a frame arrive on line 1 from Host A, the bridge concludes that Host A can be reached through the network connected to line 1. Through this process, transparent bridges build a table, which is the basis for traffic forwarding.

Transparent bridges successfully isolate intrasegment traffic, thereby reducing the traffic seen on each individual segment. This usually improves network response times as seen by the user. The extent to which traffic is reduced and response times are improved depends on the volume of intersegment traffic relative to the total traffic as well as the volume of broadcast and multicast traffic.

As the packets arrive, the bridge looks at the source address and associates it with the network address. These are then noted in a bridge address table. This table is constantly being updated with new source addresses as the network changes.

A learning bridge examines the source field of every packet it sees on each port and builds up a picture of which addresses are connected to which ports. This means that it will NOT re-transmit a packet if it knows that the destination address is connected to the same port as the bridge saw the packet on. A special problem arises if a bridge sees a packet addressed to a destination that is not in its address table. In this case the packet is re-transmitted on every port except the one it was received on.

Bridges also age address table entries, if a given address has not been heard from in a specified period of time then the address is deleted from the address table. An Example of bridge address table is shown below:

Network address	Station address
345	98765
909	87654
345	12345
345	23456
909	23567

Table 13.2 Bridge Address Table

The bridge address table is used to forward arriving packets. If an address cannot be found in the table, a discovery process is initiated. A copy of the arriving packet is sent to all network segments except the one, which sent the packet. When the destination device recognizes the address on the packet it responds to the bridge. The bridge then makes a new entry in the table noting the network and station address of this formerly unknown device. Eventually the bridge can learn the address of every device on all attached networks.

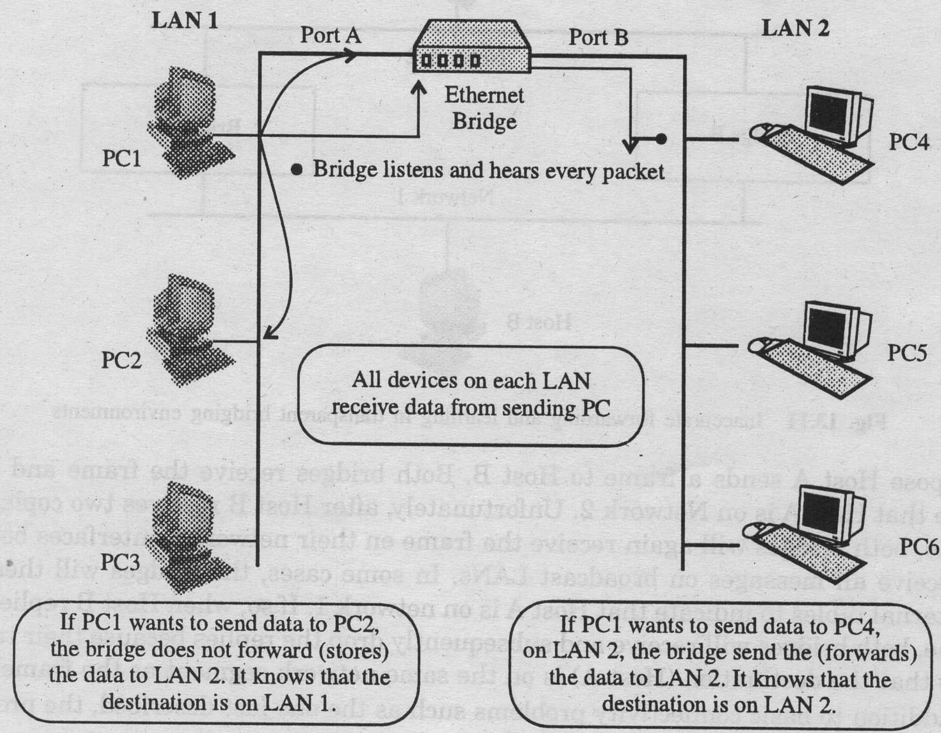

LAN 1
Port A
Port B
LAN 2
Ethernet Bridge
PC1
PC4
● Bridge listens and hears every packet
PC2
PC5
All devices on each LAN receive data from sending PC
PC3
PC6
If PC1 wants to send data to PC2, the bridge does not forward (stores) the data to LAN 2. It knows that the destination is on LAN 1.
If PC1 wants to send data to PC4, on LAN 2, the bridge send the (forwards) the data to LAN 2. It knows that the destination is on LAN 2.

Fig. 13.10

Transparent Bridges operate according to the following rules:

1. Examine all packets on all active network ports for their source address.
2. Maintain a table that tracks, which port a source address has appeared on.
3. Look up the destination addresses in this table for all packets, and if a packet's matching port is different than the port it was received on, forward the packet to the matching port.
4. If no match is found, or if the destination address is the broadcast address, forward the packet out all active ports.

This scheme is acceptable on very simple network topologies. It will not work correctly if there are multiple paths to the same destination. In this case, packets will be forwarded in a "bridging loop" which will quickly use up the available network bandwidth on the segments in question.

Bridging Loops

Without a bridge-to-bridge protocol, the transparent bridge algorithm fails when there are multiple paths of bridges and local-area networks (LANs) between any two LANs in the internetwork. Figure 13.11 illustrates such a bridging loop.

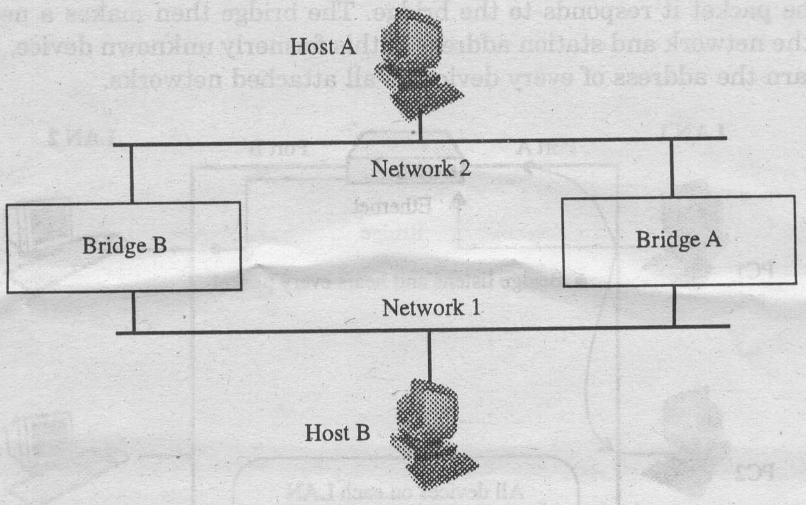


Fig. 13.11 Inaccurate forwarding and learning in transparent bridging environments

Suppose Host A sends a frame to Host B. Both bridges receive the frame and correctly conclude that Host A is on Network 2. Unfortunately, after Host B receives two copies of Host A's frame, both bridges will again receive the frame on their network 1 interfaces because all hosts receive all messages on broadcast LANs. In some cases, the bridges will then change their internal tables to indicate that Host A is on network 1. If so, when Host B replies to Host A's frame, both bridges will receive and subsequently drop the replies because their tables will indicate that the destination (Host A) is on the same network segment as the frame's source.

In addition to basic connectivity problems such as the one just described, the propagation of broadcast messages in networks with loops represents a potentially serious network problem. Referring again to Figure 13.11, assume that Host A's initial frame is a broadcast. Both bridges will forward the frames endlessly, using all available network bandwidth and blocking the transmission of other packets on both segments.

A topology with loops such as that shown in Figure 13.11 can be useful as well as potentially harmful. A loop implies the existence of multiple paths through the internetwork. A network with multiple paths from source to destination can increase overall network fault tolerance through improved topological flexibility.

2. Spanning Tree Bridges

To avoid bridging loops, an algorithm was developed which lets bridges shut off ports that provide duplicate paths to the same destination. The algorithm relies on the use of Bridge Protocol Data Units (BPDU packets), which provide information to all bridges about the "distance" in hops to each bridge port from a "root bridge". The root bridge is selected using settings entered into each bridge (with the Ethernet address acting as a tie-breaker).

Using BPDU information, a bridge can determine whether one of its ports provides an optimal path to the root bridge. If it does not, the port is shut down. If the path distance is optimal but is the same as another bridge's path, a simple protocol allows one of the ports to be shut down. In all other respects, spanning tree bridges operate in the same fashion as simple learning bridges.

Spanning tree bridges were also designed with transparency as a primary goal. A customer should be able to buy a bridge, insert it between two networks, and have everything work correctly with no hardware, software, or configuration changes on either hosts or existing bridges.

Spanning-Tree Algorithm (STA)

The spanning-tree algorithm (STA) was developed to preserve the benefits of loops while eliminating their problems. The STA designates a loop-free subset of the network's topology by placing those bridge ports that, if active, would create loops into a standby (blocking) condition. Blocking bridge ports can be activated in the event of primary link failure, providing a new path through the internetwork.

The spanning tree algorithm works by bridges interchanging special messages known as configuration bridge protocol data units. The configuration message contains enough information to enable the bridges to:

1. Elect a single bridge, from amongst all the connected bridges to be the "root" bridge.
2. Calculate the shortest path distance to the "root" bridge from each bridge.
3. For each LAN identify a "designated bridge" on that LAN that will be used for forwarding packets towards the root.
4. Choose a port on each bridge that gives the best path towards the root.
5. Select ports to be included in the spanning tree.

The effective topology after construction of the spanning tree is loop free, this is achieved by effectively choosing not to use certain links between bridges. The links are still there and may come into use if the network is re-configured.

Advantages

Easy to use. Just install the bridges. No software changes are needed.

Disadvantages

1. Does not support multi-path routing. By definition, only the bridges that belong to the spanning tree are used.
2. The path between any two hosts may not be the optimal path. An optimal path may traverse a bridge that is not part of the spanning tree and cannot be used.
3. Broadcast and multicast frames must be flooded in all cases.

Spanning tree bridges have become extremely popular in CSMA/CD networks. Vendors also offer bridges that connect LANs separated by fiber or phone links.

3. Source Routing Bridges

Source routing bridges take a completely opposite approach from spanning tree bridges:

1. They are not transparent. Hosts treat frames sent locally differently from those sent through bridges. Conceptually, the sending host specifies a road map saying which bridges the frame must go through to reach its destination.
2. Each LAN is assigned a 16-bit LAN number, and each bridge on a LAN is assigned a 4-bit bridge number. The numbers must be unique and are set by the network administrator.
3. Each frame carries a source route listing the path the frame is to take. The path consists of a sequence of {LAN number, bridge number} pairs.
4. Sending hosts (rather than bridges) chooses the source route. Host selects paths by broadcasting (flooding) special discovery frames. A discovery frame includes space for each bridge to add its number to the recorded path.
5. Eventually, a discovery frame reaches the destination host, which returns it to the sender. Each returned message contains a viable path, and the sending host chooses the shortest one.
6. Unfortunately, the discovery process leads to frame explosion. The destination may receive an exponential number of copies of the original frame.

SRB Algorithm

SRBs are so named because they assume that the complete source-to-destination route is placed in all inter-LAN frames sent by the source. SRBs store and forward the frames as indicated by the route appearing in the appropriate frame field. Figure 13.12 illustrates a sample SRB network.

Referring to Figure 13.12, assume that Host *X* wishes to send a frame to Host *Y*. Initially, Host *X* does not know whether Host *Y* resides on the same or a different LAN. To determine this, Host *X* sends out a test frame. If that frame returns to Host *X* without a positive indication that Host *Y* has seen it, Host *X* must assume that Host *Y* is on a remote segment.

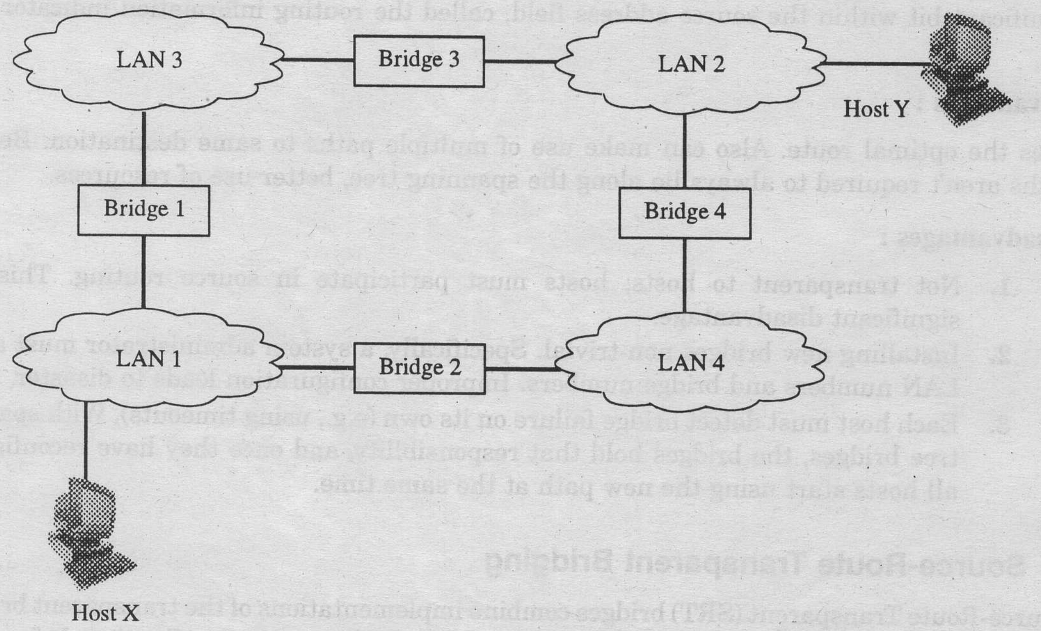


Fig. 13.12 SRB network diagram

To determine the exact remote location of Host *Y*, Host *X* sends an explorer frame. Each bridge receiving the explorer frame (Bridges 1 and 2 in this example) copies the frame onto all outbound ports. Route information is added to the explorer frames as they travel through the internetwork. When Host *X*'s explorer frames reach Host *Y*, Host *Y* replies to each individually using the accumulated route information. Upon receipt of all response frames, Host *X* chooses a path based on some predetermined criteria.

In the example in Figure 13.12, this process will yield two routes:

- LAN 1 to Bridge 1 to LAN 3 to Bridge 3 to LAN 2.
- LAN 1 to Bridge 2 to LAN 4 to Bridge 4 to LAN 2.

Host *X* must select one of these two routes. The IEEE 802.5 specification does not mandate the criteria Host *X* should use in choosing a route, but it does make several suggestions, including the following:

- First frame received.
- Response with the minimum number of hops.
- Response with the largest allowed frame size.
- Various combinations of the above criteria.

In most cases, the path contained in the first frame received will be used.

After a route is selected, it is inserted into frames destined for Host *Y* in the form of a routing information field (RIF). A RIF is included only in those frames destined for other LANs. The presence of routing information within the frame is indicated by the setting of the most significant bit within the source address field, called the routing information indicator (RII) bit.

Advantages :

Uses the optimal route. Also can make use of multiple paths to same destination. Because paths aren't required to always lie along the spanning tree, better use of resources.

Disadvantages :

1. Not transparent to hosts; hosts must participate in source routing. This is a significant disadvantage.
2. Installing new bridges non-trivial. Specifically, a system administrator must assign LAN numbers and bridge numbers. Improper configuration leads to disaster.
3. Each host must detect bridge failure on its own (e.g., using timeouts). With spanning tree bridges, the bridges hold that responsibility, and once they have reconfigured, all hosts start using the new path at the same time.

4. Source-Route Transparent Bridging

Source-Route Transparent (SRT) bridges combine implementations of the transparent bridging and the Source Routing bridging algorithms. SRT bridges use the Routing Information Indicator (RII) bit to distinguish between frames employing SRB and frames employing transparent bridging. If the RII bit is 1, a Routing Information Frame (RIF) is present in the frame, and the bridge uses the SRB algorithm. If the RII bit is 0, an RIF is not present, and the bridge uses transparent bridging.

SRT bridges are not perfect solutions to the problems of mixed media bridging. SRT bridges must still deal with the Ethernet/Token Ring incompatibilities described earlier. SRT bridging is likely to require hardware upgrades to SRBs to allow them to handle the increased burden of analyzing every packet. Software upgrades to SRBs may also be required. Further, in environments of mixed SRT bridges, transparent bridges, and SRBs, source routes chosen must traverse whatever SRT bridges and SRBs are available. The resulting paths can potentially be substantially inferior to spanning-tree paths created by transparent bridges. Finally, mixed SRB/SRT bridging networks lose the benefits of SRT bridging, so users will feel compelled to execute a complete cutover to SRT bridging at considerable expense. Still, SRT bridging permits the coexistence of two incompatible environments and allows communication between SRB and transparent bridging end nodes.

13.6 Switches

Switches are data communications devices that operate principally at Layer•2 of the OSI reference model. They are data link-layer devices that, like bridges, enable multiple physical LAN segments to be interconnected into a single larger network.

Switches are top of the line internetworking devices that employ ultra-fast memory transfer techniques to minimize segments to segments delay time. Most can operate as an ultra high-speed repeater, bridge, router or all three. Many switches even have the ability to dynamically configure what LAN segment a particular port or group of ports belongs to, regardless of the physical port to which it is attached.

Today, switching technology has emerged as the evolutionary heir to bridging-based internetworking solutions. Switching implementations now dominate applications in which bridging technologies were implemented in prior network designs. Superior throughput performance, higher port density, lower per-port cost, and greater flexibility have contributed to the emergence of switches as replacement technology for bridges and as complements to routing technology.

Upper-layer protocol transparency is a primary advantage of both bridging and switching. Because both device types operate at the link layer, they are not required to examine upper-layer information. This means that they can rapidly forward traffic representing any network layer protocol.

By dividing large networks into self-contained units, bridges and switches provide several advantages. Because only a certain percentage of traffic is forwarded, a bridge or switch diminishes the traffic experienced by devices on all connected segments. The bridge or switch will act as a firewall for some potentially damaging network errors and will accommodate communication between a larger number of devices than would be supported on any single LAN connected to the bridge. Bridges and switches extend the effective length of a LAN, permitting the attachment of distant stations that was not previously permitted.

Although bridges and switches share most relevant attributes, several distinctions differentiate these technologies. Bridges are generally used to segment a LAN into a couple of smaller segments. Switches are generally used to segment a large LAN into many smaller segments. Bridges generally have only a few ports for LAN connectivity, whereas switches generally have many. Small switches such as the Cisco Catalyst 2924XL have 24 ports capable of creating 24 different network segments for a LAN. Larger switches such as the Cisco Catalyst 6500 can have hundreds of ports. Switches can also be used to connect LANs with different media-for example, a 10-Mbps Ethernet LAN and a 100-Mbps Ethernet LAN can be connected using a switch. Some switches support cut-through switching, which reduces latency and delays in the network, while bridges support only store-and-forward traffic switching. Finally, switches reduce collisions on network segments because they provide dedicated bandwidth to each network segment.

13.6.1 Types of Switches

Switches are data link layer devices that, like bridges, enable multiple physical LAN segments to be interconnected into a single larger network. Similar to bridges, switches forward and flood traffic based on MAC addresses. Any network device will create some latency. Switches can use different forwarding techniques-two of these are stated below:

- Store-and-forward switching.
- Cut-through switching.

In *store-and-forward switching*, an entire frame must be received before it is forwarded. This means that the latency through the switch is relative to the frame size—the larger the frame size, the longer the delay through the switch.

Cut-through switching allows the switch to begin forwarding the frame when enough of the frame is received to make a forwarding decision. This reduces the latency through the switch. Store-and-forward switching gives the switch the opportunity to evaluate the frame for errors before forwarding it. This capability to not forward frames containing errors is one of the advantages of switches over hubs. Cut-through switching does not offer this advantage, so the switch might forward frames containing errors.

Many types of switches exist, including ATM switches, LAN switches, and various types of WAN switches.

ATM Switch

Asynchronous Transfer Mode (ATM) switches provide high-speed switching and scalable bandwidths in the workgroup, the enterprise network backbone, and the wide area. ATM switches support voice, video, and data applications, and are designed to switch fixed-size information units called cells, which are used in ATM communications. Figure 13.13 illustrates an enterprise network comprised of multiple LANs interconnected across an ATM backbone.

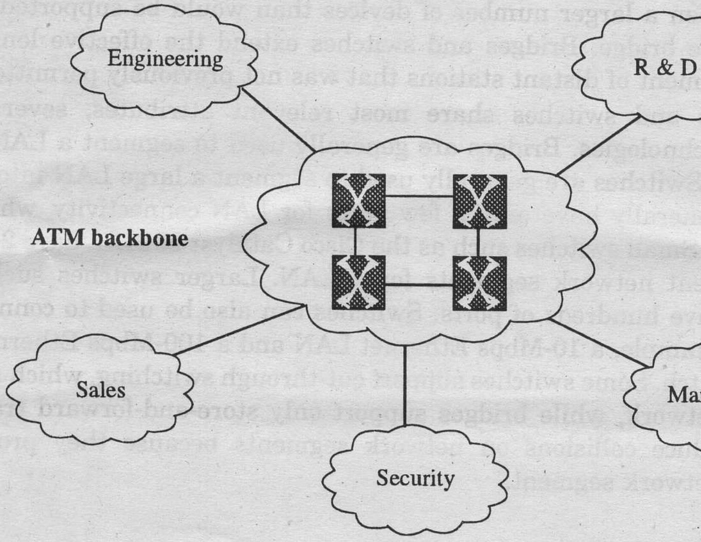


Fig. 13.13 Multi-LAN networks can use an ATM-based backbone when switching cells

LAN Switch

LAN switches are used to interconnect multiple LAN segments. LAN switching provides dedicated, collision-free communication between network devices, with support for multiple simultaneous conversations. LAN switches are designed to switch data frames at high speeds.

Figure 13.14 illustrates a simple network in which a LAN switch interconnects a 10-Mbps and a 100-Mbps Ethernet LAN.

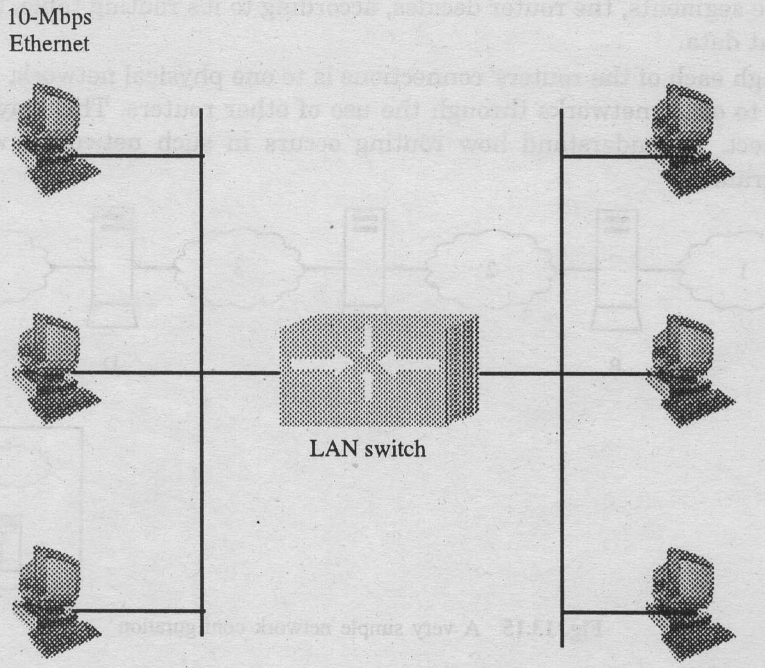


Fig. 13.14 A LAN switch can link 10-mbps and 100-mbps ethernet segments

13.7 Routers

Routers are internetworking devices that operate at the third layer of the OSI model—the Network layer. This makes their operation somewhat different from that of repeaters and MAC (Media Access Control) bridges. Both of these devices are used to connect networks that are identical with respect to the media access scheme used. Therefore, information was transferred based on MAC addresses. This also made these devices protocol independent-specifically, they are independent of the protocols working above the MAC layer.

Routers are able to provide connectivity for mixed MAC environments by working with a higher-layer protocol. This permits the connection of network segments that are different. However, it must be noted that the networks to be interconnected share that same protocol stack-routers are not capable of protocol translation. Most routers are multiprotocol routers and are capable of handling multiple Network layer protocols-the router must be equipped with appropriate software for each protocol to be supported.

Unlike bridges, routers can intelligently determine the most efficient path to any destination, based on predetermined delimiters.

In a heterogeneous environment, such as networks, a need of connection devices which would inter-connect two different technologies is essential. In this environment the router is that device. As it's name implies, the router also serves as a routing switch-board. Routers connect two or more networks and forward data packets between them. When data arrives from one of the segments, the router decides, according to it's routing table, to which segment to forward that data.

Even though each of the routers' connections is to one physical network, that one network could connect to other networks through the use of other routers. This way, many networks can interconnect. To understand how routing occurs in such networks, we'll refer to the following diagram:

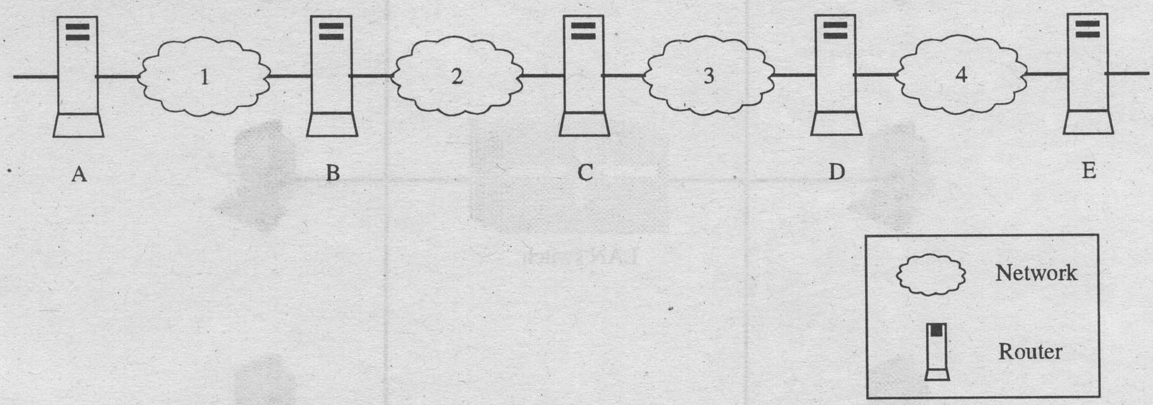


Fig. 13.15 A very simple network configuration

In the above configuration, router *B*'s table would say, for instance, that data going to network 4 should go to router *C*. Packets going from network 1 to network 4 would go through router B into network 2 and so on, till they reach their destination.

We would like to emphasize that routers know only about networks, and not about hosts. In IP networks, routers utilize the fact that each host's IP address contains two parts: the host's network address, and the host's number on that network. Routers examine the data destination address, extract, from it, the target network address, and decide, based on this network address, where to transfer the data.

A router is actually a special computer that is dedicated to the task of interconnecting networks. It moves information from its source to its destination regardless of the middleware. A router resembles a bridge (they both have conventional processor, memory and few different I/O interfaces, each for another network it connects), but while bridging occurs at the link layer, routing occurs at the network layer.

13.7.1 Router Responsibilities

The role of the router is to direct packets along the most efficient, economical route in mesh networks where there are many redundant paths from the source device to the destination device. Although some Network Operating Systems support routing by the file server, most routers are stand-alone devices. The routing function tends to slow the server performance. The throughput of a router is largely determined by its internal components and architecture.

Routers have two major responsibilities that are stated below:

1. **Optimizing the routing paths :** A router uses a routing algorithm to determine the optimal path to the destination. These algorithms maintain routing tables that contain route information such as destination/next hop association. Selecting a communications path through the network is a two-step process. The first step is to create and maintain a routing table and the second, to select the best path for the next step of the packet's journey. This second step is based on information found in the packet and in the routing table. Both routing tables and selection of pathways through the network form the basis of how routers function.
2. **Switching-transport of packets over networks :** When computer wants to send a packet over the net, it formats a packet with the router's physical address and the destination address (protocol address) of the target host. The router searches its routing tables for the destination host. If there is no entry for the destination host the router usually drops the packet, otherwise (there is an entry for the destination host) it replaces the physical address with the next hop's address and retransmits the packet. The next hop isn't necessarily the ultimate destination host; it may be another router that performs the same routing again. A packet may visit few routers/hosts on its route, each time it's destination physical address changes.

13.7.2 Routers Versus Bridges

A problem commonly faced in internetworking concerns the choice between bridges and routers. The current trend is the development of devices providing the functionality of both bridges and routers. Until a single such unit is available, a choice will need to be made. The following table summarizes some characteristics of bridges and of routers.

Characteristic	MAC Bridge	Router
Network dependence	Links same-type LANs	Links all forms of LANs
Protocol dependence	Protocol independent at the LLC layer and above	Network layer protocols must be common
Bandwidth utilization	Fixed by the Spanning Tree Algorithm or Source Routing	Optimized for traffic loading across the entire network
Data flow control	Access limitation and filtering according to MAC address	Network address dependent; can exert flow control
WAN access	Usually single port	Possible to construct a complex mesh of point-to-point links
Management	Statistics about frames and bridge operations	Statistics about specific protocols
Performance	May suffer in heavy traffic environments	Similar to bridges but can limit traffic from sources by quenching

TABLE 13.3 Router/Bridge Comparison

Combined Bridge And Router Architectures

The distinction between bridges and routers is not as clear as it once was. Over time many intermediate devices have been introduced, including bridging routers, routing bridges, bridge routers, router bridges and brouters. What all of these devices have in common is that they perform some functions associated with bridges and some functions associated with routers.

Bridge routers and routing bridges are considered to be the same. They are essentially bridges that provide some routing functions. Router bridges and bridging routers are also considered to be the same. They are routers that perform some bridging functions. Brouters are considered to be true hybrid devices. Through the use of software, these devices can be reconfigured to function as a bridge, a router or to provide for a mix of the two functions.

Bridge/Router Systems

Bridge/router systems have their origins in the bridge environment. Often, this type of device is capable of supporting only one type of network technology—either all Ethernet or all Token-ring. Bridging-type functions dominate these devices but they are capable of routing specific Network layer protocols. Bridging performance is usually at full bridge speeds but routing tends to be done at lower speeds. Most can also be interconnected to standardized bridges and routers. Since these devices are bridge-based, most tend to be two-port devices with single processor architecture.

Router/Bridge Systems

Router/bridge systems have their origins in the router environment. These types of devices were of benefit in router-based networks where there were certain types of network traffic that could not be routed—they lacked a Network layer protocol. Using a router with some bridging capabilities allowed these unroutable protocols to be bridged instead. Routing-type functions predominate with bridging used most to transfer those protocols, which cannot be routed. The performance of these devices for both the routing and bridging functions is at full speed. Most can also be interconnected to standardized routers and bridges. These devices tend to be multiple port devices with each port supported by its own processor.

13.8 Brouters

A brouter can work in either the second and third layers of the OSI model—the data link layer or the network layer. Brouters combine the functions of a bridge and a router. If it can't route a packet, it acts as a bridge. Brouters are particularly useful if you have two or more different networks. Working as a bridge, a brouter is protocol independent and can be used to filter local area network traffic. Working as a router, a brouter is capable of routing packets across networks.

Brouters have an independent origin-they were not developed by either bridge or router manufacturers. Modern brouters are used primarily in new network installations where the functionality of both a bridge and a router is required. An advantage of brouters is that they can be readily adapted to new demands. Brouters are capable of functioning solely as bridges, solely as routers, or as true hybrid devices. However, they are usually proprietary devices that can be interconnected only to certain other devices. Full connectivity to standard bridges and routers is not always possible.

Bridging performance is usually at full bridging speeds while routing speeds tend to be slower. These devices tend to be software configurable and are supported by a single processor architecture.

13.9 Gateways

Gateways work at OSI model layer 7-the application layer. Gateways are special devices designed to overcome incompatibilities between networks. They are typically designed to perform a specific purpose (i.e., a system network architect (SNA) gateway) and are sometimes slow and expensive. However, they form an essential part of the internetworking system in situations where it is not practical or possible to use a single networking technology.

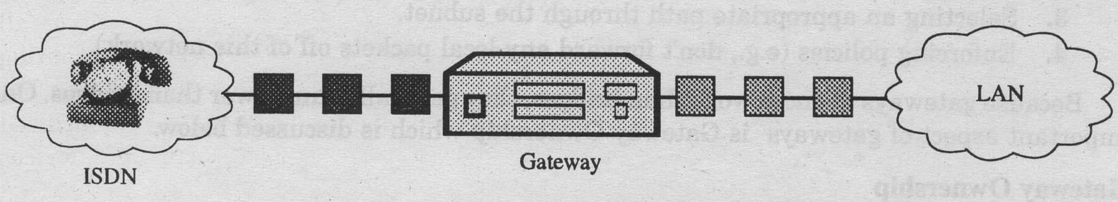


Fig. 13.16 Gateway connecting two dissimilar networks

Gateway is a term that once used to refer to a routing device. Today, in the TCP/IP world, the term router is used to describe such a device. The term gateway now refers to special-purpose devices that perform protocol conversions. Gateways implement application layer conversions of information received from various protocols.

A gateway functions to reconcile differences between two dissimilar networks. Messages are not only repackaged for transmission between different networks (CSMA/CD to token-passing), but the contents of the messages are converted into a format the destination can use and understand. A gateway is generally a dedicated computer with an interface card and at least some type of software for both of the environments being connected. The gateway then runs special software that provides the necessary conversion and translation services, which, in turn, allow the two environments to communicate.

The role of the gateway is to overcome problems associated with protocol incompatibility. Gateways are available as stand-alone devices or in the form of a network station functioning as a gateway server. In both cases, the gateway requires appropriate network adapters—LAN, WAN or both—and appropriate software. It is the software, which is responsible for translating the messages from the received format to the format understood by the destination system.

In the case of bridges and repeaters, their functionality was largely independent of the environments they interconnected. However, gateways are very dependent on the systems they are interconnecting. Each type of system requires a specific type of gateway. Therefore, an individual type of gateway has a limited range of capabilities.

Another major difference between gateways and other internetworking devices is the level of compatibility with respect to the OSI model. Repeaters, bridges and routers have little concern for the contents of the packets they transport. Gateways require compatibility at the uppermost layers of the OSI model. They are application-specific-gateways providing interconnection between electronic mail systems cannot be used for general file transfer purposes.

Examples of gateways found on todays markets are:

- **VocalTec Gateway :** A gateway that converts human speech traveling on analog phone lines into local area network protocol data, and vice-versa.
- **RadVision Gateway :** Converts video from digital phone lines into local area network protocol data, and vice-versa.

Gateways provide increased flexibility compared to bridges in terms of:

1. Translating addresses between dissimilar networks.
2. Fragmenting large packets for transmission across networks that carry only small maximum packet lengths.
3. Selecting an appropriate path through the subnet.
4. Enforcing policies (e.g., don't forward any local packets off of this network).

Because gateways do more work than bridges, they generally run slower than bridges. One important aspect of gateways is Gateway Ownership which is discussed below.

Gateway Ownership

One issue that arises with gateways is who owns them. Typically, bridges connect LANs of one organization, and the issue does not arise there. The ownership question is important because someone has to be responsible for the gateway's operation and dual ownership frequently leads to finger pointing when something goes wrong.

One solution is to use half gateways. If two countries are involved, for instance, each country owns its half of the gateway, with a wire separating the two. A special protocol operates over the wire, and each half of the gateway is responsible for implementing the protocol.

Responsibility Of Gateways

The primary functional responsibility of a gateway is to translate from one protocol stack to another. There are three ways in which a gateway may operate. These are as follows:

- **Remote protocol encapsulation :** In this case, the data from the remote system is encapsulated by the protocols used by the local system. The encapsulation is stripped away by the local system protocol stack as the data passes from the bottom layer to the upper layer. The application layer of the local protocol stack receives the data in its native format.
- **Local protocol encapsulation :** This is similar to remote protocol encapsulation, except the data from the local system is encapsulated by the protocols used by the remote system.
- **Protocol conversion :** In the case of protocol conversion, the gateway is responsible for reorganizing the data content. The reorganization places appropriate information in the correct position in the message. Protocol conversion is the process by which the data format used in one communications system is translated into the data format

used by another system. There are multiple levels at which protocol conversion may take place.

Some forms of encapsulation are supported by bridges and routers. However, gateways are the only devices capable of supporting both encapsulation and protocol conversion.

Selection of Gateways

Once the need for a gateway has been established, the next step is to decide the type of gateway that is needed. Some of the issues to consider when making a selection include the following:

- The types of network environments that need to be supported-LAN-to-LAN, LAN-to-WAN and/or WAN-to-WAN.
- The identification of the protocols being used and the protocols that will require translation.
- The number of network links that need to be supported.
- Whether or not routers will be part of the internetworking infrastructure.
- If any host computer systems need to be supported, and if so which one. Special interface boards may be required.
- The number of concurrent sessions that need to be supported by the gateways.
- The amount of buffer storage area needed to compensate for transmission speed mismatches.

● ● ●

14

Transport Layer

Introduction

The transport layer is the fourth layer of the OSI reference model. It provides transparent transfer of data between end systems using the services of the network layer (e.g., IP) below to move PDUs of data between the two communicating systems.

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

Under normal conditions, the transport layer creates a distinct network connection for each transport connection required by the session layer. If the transport connection requires a high throughput, however, the transport layer might create multiple network connections, dividing the data among the network connections to improve throughput. On the other hand, if creating or maintaining a network connection is expensive, the transport layer might multiplex several transport connections onto the same network connection to reduce the cost. In all cases, the transport layer is required to make the multiplexing transparent to the session layer.

The transport layer is a true source-to-destination or end-to-end layer. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. Many hosts are multi-programmed, which implies that multiple connections will be entering and leaving each host. There needs to be some way to tell which message belongs to which connection. The transport header is one place this information could be put.

In addition to multiplexing several message streams onto one channel, the transport layer must take care of establishing and deleting connections across the network. This requires some kind of naming mechanism, so that process on one machine has a way of describing with whom it wishes to converse. There must also be a mechanism to regulate the flow of information, so that a fast host cannot overrun a slow one. Flow control between hosts is distinct from flow control between switches, although similar principles apply to both.

The transport layer is important as it segments the data from the upper layers, and passes these onto the network layer, which adds the network address to the segments (these are then called packets). The network layer allows for delivery of these segments at the receiver. Once delivered the transport layer then takes over and reassembles the data segments back into a form that can be delivered to the layer above. These services are often known as end-to-end services, as the transport layer provides a logical connection between two end points on a network.

The transport service is said to perform "peer to peer" communication, with the remote (peer) transport entity. The data communicated by the transport layer is encapsulated in a transport layer PDU and sent in a network layer SDU. The network layer nodes (i.e., Intermediate Systems (IS)) transfer the transport PDU intact, without decoding or modifying the content of the PDU. In this way, only the peer transport entities actually communicate using the PDUs of the transport protocol.

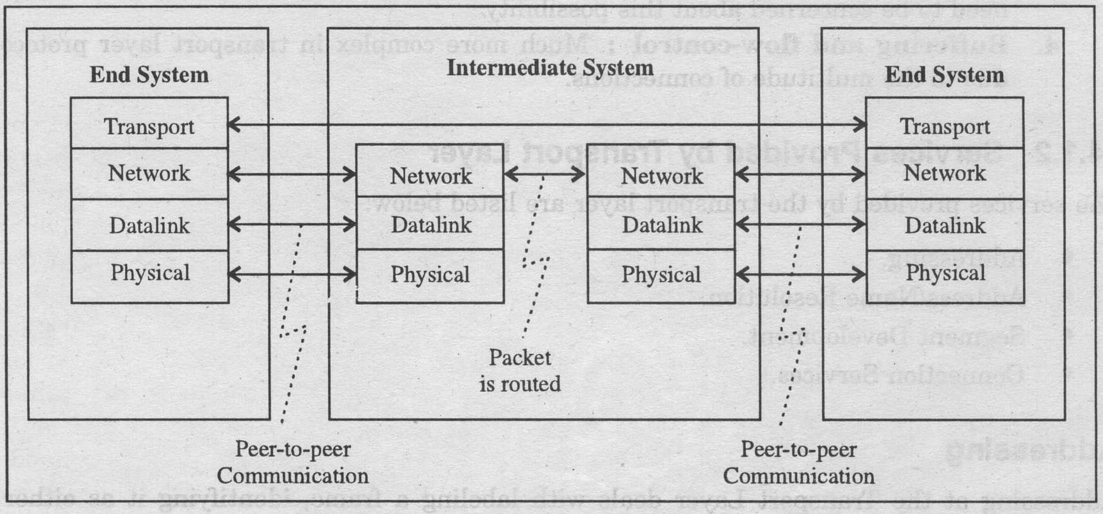


Fig. 14.1 Two end systems connected by an intermediate system (in this case a router)

14.1 Transport Layer Characteristics

- Transport Layer protocols reside in the source and destination nodes.
- Transport Layer provides a consistent service interface between applications and the network.
- Transport protocol normally included in operating system.
- Layers above transport are linked into user program.
- Transport Layer must provide reliable connection despite

- □ Packet loss due to best-effort connectionless network services or due to Virtual Circuit resets on "reliable" connection-oriented network Services
- □ Packet reordering that can occur with connectionless service or when a transport Connection is split across multiple Virtual Circuits

14.1.1 Transport Layer vs. DLL Layer

Transport Layer and Data Link Layer bear some resemblance: Both perform error control, resequencing, and flow control. Data Link layer operates on a link basis. Transport Layer operates end-to-end across a network.

Differences between datalink layer and transport layer protocols

1. **Addressing :** Datalink layer protocols don't need to deal with addresses. Transport layer protocols explicitly need addressing information.
2. **Connection establishment :** Datalink layer needs no such procedure. Initial connection establishment is elaborate in transport layer protocols.
3. **Subnet as storage :** Datalink layer protocols don't need to be concerned about a packet being (temporarily) stored at an intermediate router. Transport layer protocols need to be concerned about this possibility.
4. **Buffering and flow-control :** Much more complex in transport layer protocols due to the multitude of connections.

14.1.2 Services Provided by Transport Layer

The services provided by the transport layer are listed below:

- Addressing.
- Address/Name Resolution.
- Segment Development.
- Connection Services.

Addressing

Addressing at the Transport Layer deals with labeling a frame, identifying it as either a connection or a transaction.

Connection Identifiers—A connection identifier is also known as a port or socket. It labels each frame by conversation so the receiving device knows which process it has been transmitted from. This is for keeping track of multiple-message conversations.

Transaction Identifiers—They are used when frames sent are simple requests or responses. They are considered one-time events, and no multiple message conversations are tracked, even though there may be one going on.

Address/Name Resolution

Each network address is a binary number, often 32-bits. Keeping track of these addresses is near impossible for people wanting to transmit over a network. Therefore with some protocols, each address is assigned a name in recognizable words. A service on the network, (either at each device or by a central name server) translates the word addresses into numeric addresses. There are *two methods* of Address/Name Resolution:

Service-Requester-Initiated Address/Name Resolution—With this method, a device that wishes to request services broadcasts a packet requesting information for a name, address or service. The device corresponding to the name, address, or service sends back the required information needed to complete a transaction.

Service-Provider-Initiated Address/Name Resolution—This method consists of a central directory server (name server) that collects name, address and service information. The service providers broadcast this information periodically and devices requesting services request address information from the name server.

Segment Development

The Transport Layer is where incoming and outbound messages are broken down or recombined to fit the required size format. It also combines multiple small messages into a single segment to improve network efficiency when the messages are smaller than maximum allowable size. A connection identifier (CID) identifies each component. The CID enables the transport layer of the receiving device to deliver each message to the proper process.

Connection Services

The Transport Layer can also perform connection services such as:

- **Segment Sequencing**—When large messages are divided, the Transport Layer reorders them once received before reassembling them into a message.
- **Error Control**—To prevents errors from lost or duplicate segments, the Transport Layer enables the following strategies:
  - Unique segment sequence numbers.
  - Virtual circuits, permitting only one virtual circuit per session.
  - Timeouts removed from the network segments that have been misrouted and have remained on the network past a specified time.
  - Corruption is also detected by end-to-end error control by way of checksums.
- **End-to-End Flow Control**—Acknowledgements are used to manage end-to-end flow control. There are negative acknowledgements as well as go back *n* or selective repeat acknowledgements.

  Go back n—Request retransmission of packets starting at *n*.
  Selective Repeat—Request specific packets to be retransmitted.

14.1.3 Elements of Transport Protocols

- **Connection Management**—Establish, refuse and release connections. Manage mapping between transport and network connections or build a connection-oriented service from a connectionless network service.

- **Fragmentation and reassembly of application data**—Fragment data stream into packet sizes suitable for the network. Where multiple transport connections exist packets must be correctly associated with each connection. Receiver must reassemble packets and resequence the data stream.
- **Recovery from network failures**—Reassignment after network disconnects and resynchronization after network resets.
- **Error Control and Flow Control**—Especially for LANs.
- **Multiplexing**—Mapping multiple Transport Connections to one Network Connection.
- **Splitting and Recombining**—Mapping one Transport Connection to multiple Network Connections.

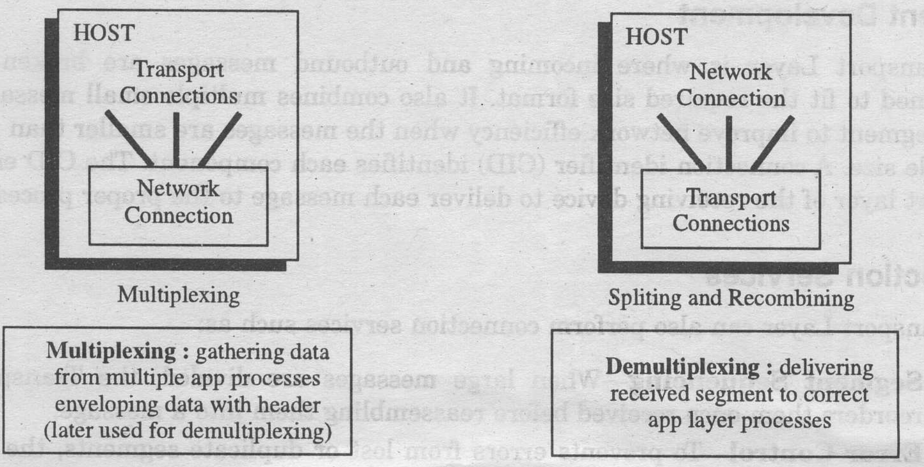

HOST
Transport Connections
Network Connection
Multiplexing
HOST
Network Connection
Transport Connections
Spliting and Recombining
Multiplexing : gathering data from multiple app processes enveloping data with header (later used for demultiplexing)
Demultiplexing : delivering received segment to correct app layer processes

Fig. 14.2 Multiplexing and demultiplexing

14.1.4 Functions of the Transport Layer

The great trick of the transport level is to allow multiple applications to communicate over a network, at the same time. This is achieved by identifying each connection with a unique value (a socket number) and adding sequence numbers on each of the segments. Segments can also be sent to many different destinations (using the network layer to identify the destination).

- **Process-Level Addressing :** Addressing at layer two deals with hardware devices on a local network and layer three addressing identifies devices on a logical internetwork. Addressing is also performed at the transport layer, where it is used to differentiate between software programs. This is part of what enables many different software programs to use a network layer protocol simultaneously. The best example of transport-layer process-level addressing is the TCP and UDP port mechanism used in TCP/IP, which allows applications to be individually referenced on any TCP/IP device.

- **Multiplexing and Demultiplexing :** Transport layer protocols on a sending device multiplex the data received from many application programs for transport, combining them into a single stream of data to be sent. The same protocols receive data and then demultiplex it from the incoming stream of datagrams and direct each package of data to the appropriate recipient application processes.
- **Segmentation, Packaging and Reassembly :** The transport layer segments the large amounts of data it sends over the network into smaller pieces on the source machine and then reassemble them on the destination machine. This function is similar conceptually to the fragmentation function of the network layer; just as the network layer fragments messages to fit the limits of the data link layer, the transport layer segments messages to suit the requirements of the underlying network layer.
- **Connection Establishment, Management and Termination :** Transport layer connection-oriented protocols are responsible for the series of communications required to establish a connection, maintain it as data is sent over it and then terminate the connection when it is no longer required.
- **Acknowledgments and Retransmissions :** As mentioned above, the transport layer is where many protocols are implemented that guarantee reliable delivery of data. This is done using a variety of techniques, most commonly the combination of acknowledgments and retransmission timers. Each time data is sent a timer is started; if it is received, the recipient sends back an acknowledgment to the transmitter to indicate successful transmission. If no acknowledgment comes back before the timer expires, the data is retransmitted. Other algorithms and techniques are usually required to support this basic process.
- **Flow Control :** Transport layer protocols that offer reliable delivery also often implement *flow control* features. These features allow one device in a communication to specify to another that it must "throttle back" the rate at which it is sending data, to avoid bogging down the receiver with data. These allow mismatches in speed between sender and receiver to be detected and dealt with.

Transport Protocol Data Units

Data is exchanged between Transport peers in TPDU's. TPDU's are of course encapsulated within packets for transmission over the network. The protocols that are used by the Transport layer are essentially the same as underlying layers, e.g., TPDU's require sequence numbers for flow and error control.

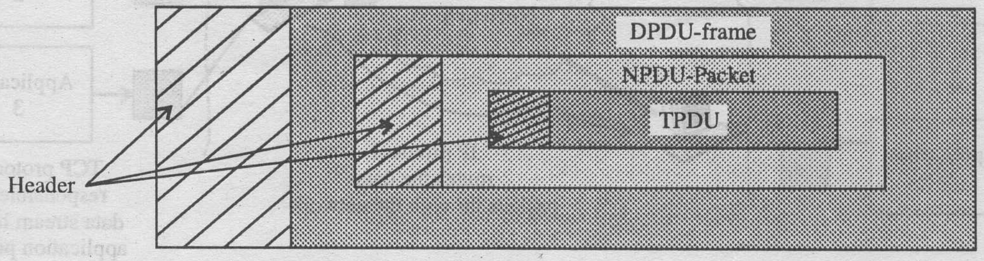


Fig. 14.3 TPDU's encapsulated within packets for transmission

TPDU's may be given the common name, *messages*. They may well be fragmented by the Network layer but this will be transparent to the Transport layer.

14.1.5 Transport Layer Protocols

At any time, a host may be using several network connections to various services on other hosts. IP allows packets to be routed between hosts on an Internet, but once a packet reaches its destination host, a mechanism is needed whereby it can be routed to the appropriate application. IP addresses themselves do not provide sufficient information to do this.

A solution to the problem is to use a higher-level protocol for this task, while still using IP as the underlying delivery mechanism. The simplest such higher-level protocol is the User Datagram Protocol (UDP). UDP datagrams consists of a header and zero or more bytes of actual data. The header has four fields: the source and destination ports, a data integrity checksum whose use is optional and the length of the datagram. The 'ports' are 16-bit numbers used to identify the connection end points on either side. A particular network connection is then identified by a four-tuple (source address, source port, destination address, destination port).

UDP is still connectionless and unreliable. To support connection-oriented applications which require reliable, in-sequence data streams, the Transport Control Protocol or TCP is used. Like UDP, TCP also uses ports to identify connection endpoints. Checksum usage in TCP is mandatory to ensure reliability. TCP headers include sequence numbers representing a byte offset in the data stream; these sequence numbers are used on the receiving end to arrange data received in TCP packets in the correct order. TCP makes use of acknowledgments to inform the sender that data has arrived correctly at the recipient (an acknowledgment consists of the sequence number of the next expected packet).

The most important transport protocol is TCP, which typically uses IP as an addressing scheme. TCP and IP (TCP/IP) are the standard protocols that are used on the Internet and

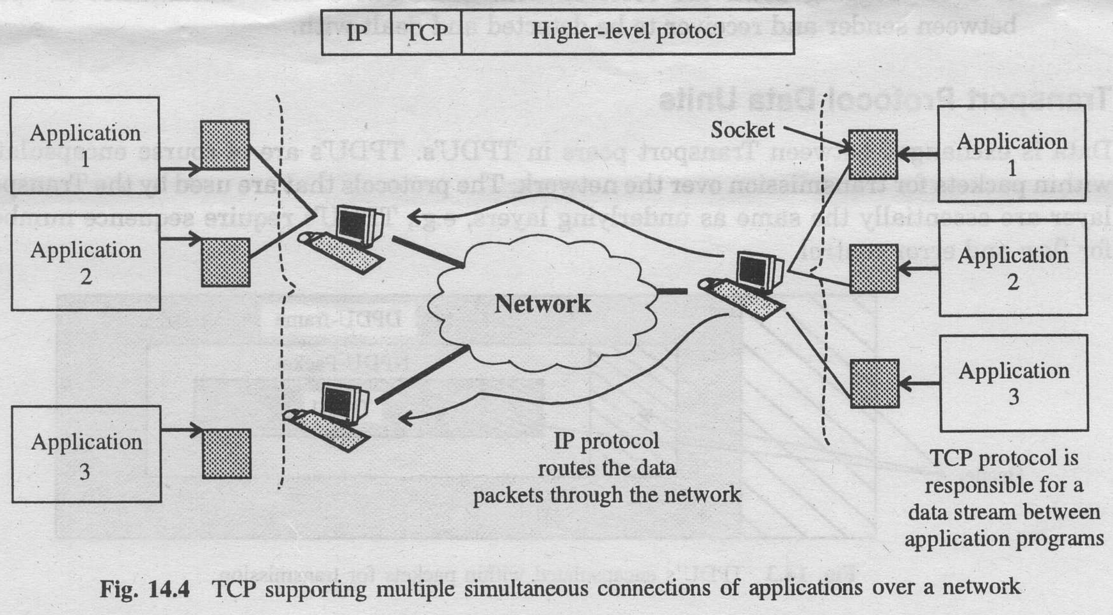

IP
TCP
Higher-level protocl
Application 1
Application 2
Application 3
Socket
Network
Application 1
Application 2
Application 3
IP protocol routes the data packets through the network
TCP protocol is responsible for a data stream between application programs

Fig. 14.4 TCP supporting multiple simultaneous connections of applications over a network

also on UNIX networks. TCP works well because it is simple, but reliable and it is an open system where no single vendor has control over its specification and its development. An important concept of TCP/IP communications is the usage of ports and sockets. A port identifies a process type (such as FTP, TELNET and so on) and the socket identifies a unique connection number. In this way, TCP/IP can support multiple simultaneous connections of applications over a network as shown in fig 14.4.

Before discussing the protocols in detail let us understand what are ports.

14.2 Ports

There are two meanings in common use for the term "port", one having to do with hardware and the other related to software ports. On computer and telecommunication devices (hardware) a port is generally a specific physical location for connecting to some other device. Usually this involves a socket and plug. Generally, there are one or more serial ports and one parallel port on a personal computer. The serial port supports sequential, bit-by-bit transmission to peripheral devices such as scanners. A parallel port supports multiple bit transmission to devices like printers. Newer computers may also employ USB (universal serial bus) ports. USB connections move data at up to 12 mbps, about 100 times faster than a typical serial port and more than four times faster than a parallel port.

This article is concerned with the second or software, type of port used in Internet communications. In software usage, a port is a logical, rather than physical, connection. When using the Internet communications protocol, transmission control protocol/Internet protocol (TCP/IP), designating a port is the way a client program specifies a particular server program on a computer in a network. Basically, a port number is a way to identify the specific process to which an Internet or other network message is to be forwarded when it arrives at a server.

Both TCP and UDP use ports to exchange information with applications. A *port* is an extension of an address, similar to adding an apartment or room number to a street address. A letter with a street address will arrive at the correct apartment building, but without the apartment number, it will not be delivered to the correct recipient. Ports work in much the same way. A packet can be delivered to the correct IP address, but without the associated port, there is no way to determine which application should act on the packet. Once the ports have been defined, it is possible for the different types of information that are sent to one IP address to then be sent to the appropriate applications. By using ports, a service running on a remote computer can determine what type of information a local client is requesting, can determine the protocol needed to send that information, and maintain simultaneous communication with a number of different clients.

For example, if a local computer attempts to connect to the website www.gnduonline.org, whose IP address is 92.70.122.203, with a web server running on port 70, the local computer would connect to the remote computer using the socket address : 92.70.122.203 : 70.

Higher-level applications that use TCP/IP such as the web protocol and hypertext transfer protocol (HTTP) use ports with pre-assigned numbers. These are well-known ports, to which numbers have been assigned by the Internet Assigned Numbers Authority. Some application processes are given port numbers dynamically when each connection is made.

When a server program is started via a port connection, it is said to bind to its designated port number. When another client program wants to use that server, it also must send a request to bind to the designated port number. Ports are used in TCP to name the ends of logical connections that carry long-term conversations. To the extent possible, these same port assignments are used with UDP.

Port numbers can be changed via software settings or can be remapped internally using routers that can change header information. Headers provide the basis for synchronizing communications sessions, control of errors, routing and firewall management. The following table provides a description of a typical header.

Data Link/ Physical Header	IP Address/Header	Port Address	Data
Locates the correct device on the local area network (network interface-hardware layer of TCP/IP)	Locates the correct device in a network using TCP/IP (internetworking layer of TCP/IP)	Locates the correct application/service (transport layer of TCP/IP)	Data

Table 14.1: Internet Addressing–TCP/IP

When sending information, each layer in the protocol stack adds a header to the data. Conversely, at the receiving end, each layer removes a header. A connection in TCP/IP might look like this (see explanation of numbering below the example):

185.15.4.1,	1160,	185.15.4.3,	23,	TCP
(1)	(2)	(3)	(4)	(5)

1. Source or client IP address. The final segment is the host identifier.
2. Source or client port that would be randomly assigned.
3. Destination or server IP address (here, because the first three segments are the same as the source address, this is the same network; different host).
4. Destination or server port. In this case, port 23 for a Telnet application/request.
5. TCP protocol.

Any address scheme can be used for an isolated network with some restrictions depending on the type of network.

14.2.1 Classification of Ports

Port numbers are divided into three ranges: the well-known ports, the registered ports and the dynamic and/or private ports.

The well-known ports—0 through 1023 : The well-known ports are assigned by the Internet Assigned Numbers Authority (IANA) and are typically used by system-level or root processes. Well-known applications running as servers and passively listening for connections typically use these ports. Some examples include: FTP (21), TELNET (23), SMTP (25) and HTTP (80).

The registered ports—1024 through 49151 : Listed by the IANA and on most systems can be used by ordinary user processes or programs executed by ordinary users. Registered ports are typically used by end user applications as ephemeral source ports when contacting servers, but they can also identify named services that have been registered by a third party.

The dynamic and/or private ports—49152 through 65535 : Not listed by IANA because of their dynamic nature. Dynamic/private ports can also be used by end user applications, but are less commonly so. Dynamic/private ports do not contain any meaning outside of any particular TCP connection. There are 65535 possible ports officially recognized.

Socket

A socket is a network communications endpoint. The analogy is to a wire (the network data connection) being plugged into a socket. Sockets come in two primary types:

- An active socket is connected to a remote active socket via an open data connection. Closing the connection destroys the active sockets at each endpoint.
- A passive socket is not connected, but rather awaits an incoming connection, which will spawn a new active socket.

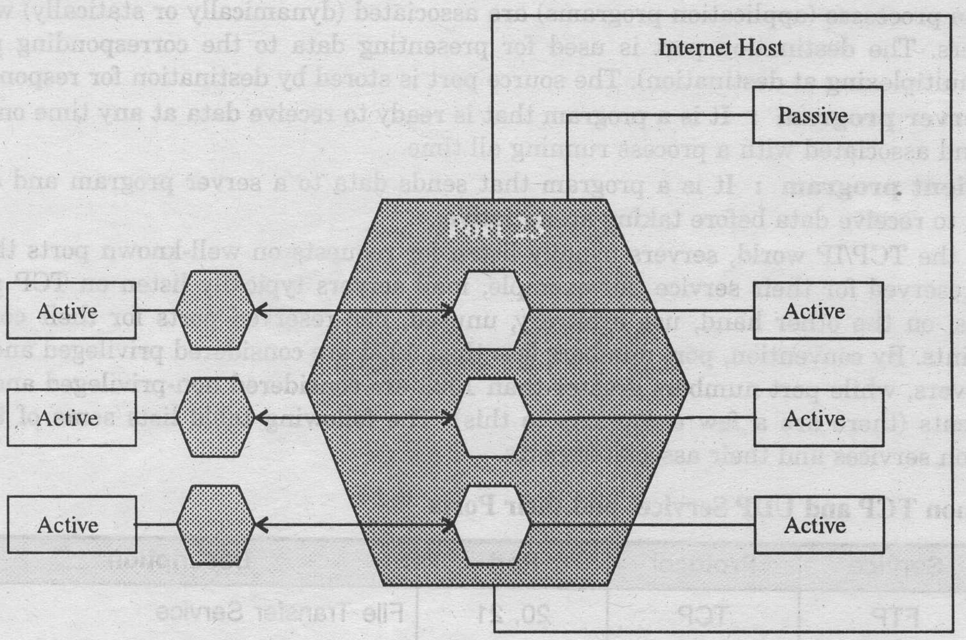


Fig. 14.5 A port having single passive socket and multiple active sockets

A socket is not a port, though there is a close relationship between them. A socket is associated with a port, though this is a many-to-one relationship. Each port can have a single passive socket, awaiting incoming connections, and multiple active sockets, each corresponding to an open connection on the port as is illustrated in figure 14.5.

14.3 Client Server Model

We have covered the underlying network transport mechanism and can now consider how distributed applications are built on top of this. Almost all network-based applications are based on a client-server model of interaction. A server is a program that offers a service to other programs on the same or other hosts on the network. A client is a program that requests service from a server. Servers usually run continuously and can typically service multiple requests from multiple clients, often concurrently. Clients, on the other hand, usually have a limited lifetime.

For Transport layer connection to be established a process must be listening to a Transport address and a second process must make a connection request. Typically processes don't listen to Transport addresses. A process must register with the operating system to "bind" to a Transport address. The process that is listening is called a server. The process that is making the connection request is called a *client*. For example the telnet program is a client program. It can be used to connect to a given process by specifying the host address and transport address.

The processes (application programs) are associated (dynamically or statically) with port numbers. The destination port is used for presenting data to the corresponding program (= demultiplexing at destination). The source port is stored by destination for responses.

Server program : It is a program that is ready to receive data at any time on a given port and associated with a process running all time.

Client program : It is a program that sends data to a server program and does not expect to receive data before taking an initiative.

In the TCP/IP world, servers usually listen for requests on well-known ports that have been reserved for their service (for example, mail servers typically listen on TCP port 25). Clients, on the other hand, use arbitrary, unused, non-reserved ports for their connection endpoints. By convention, port numbers less than 1024 are considered privileged and for use by servers, while port numbers greater than 1023 are considered non-privileged and for use by clients (there are a few exceptions to this). The following table lists some of the most common services and their associated ports.

Common TCP and UDP Services and their Ports

Service	Protocol	Port	Description
FTP	TCP	20, 21	File Transfer Service
telnet	TCP	23	Virtual terminal service
SMTP	TCP	25	Electronic mail transfer
DNS	TCP/UDP	53	Domain name service
TFTP	UDP	69	Trivial file transfer
HTTP	TCP	80	World wide web hyper text transfer
POPS	TCP	110	Electronic mail retrieval
SNMP	UDP	161, 162	Network management service

Generic Network Server Algorithm

```
start listening for requests on the well known port
loop forever
begin
          if a request is received from some client
          then create a slave process to handle that
                                     client's request
end
```

Servers usually operate according to this general algorithm. By running a slave process to handle each client request, the server can deal with multiple concurrent requests.

Process Server

To prevent a large number of sleeping servers, a process server can bind to a number of different Transport addresses and hand-over an incoming connection request to an actual server of the requested type. An example is illustrated in the figure 14.6.

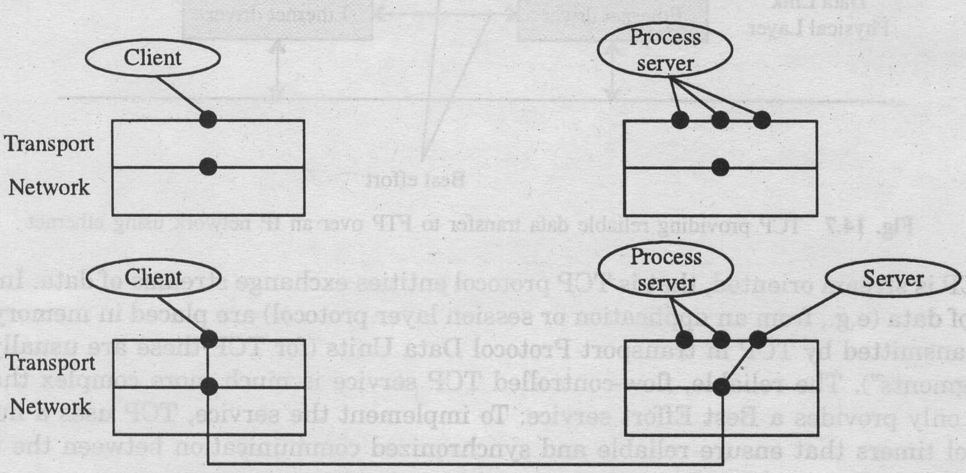


Fig. 14.6 A process server that hands over a connection request to the actual server

14.4 Transmission Control ProtocoL (TCP)

The **Transmission Control Protocol (TCP)** is one of the core protocols of the Internet protocol suite. Using TCP, programs on networked computers can create connections to one another, over which they can send data. The protocol guarantees that data sent by one endpoint will be received in the same order by the other, and without any pieces missing. It also distinguishes data for different applications (such as a Web server and an email server) on the same computer.

TCP supports many of the Internet's most popular applications, including HTTP and SMTP.

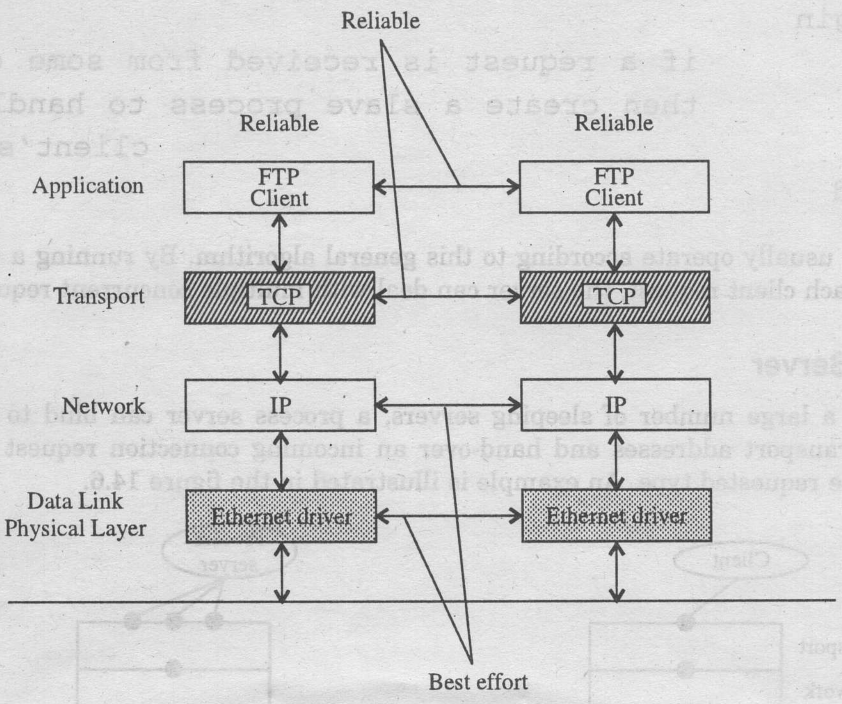


Fig. 14.7 TCP providing reliable data transfer to FTP over an IP network using ethernet

TCP is stream oriented, that is TCP protocol entities exchange streams of data. Individual bytes of data (e.g., from an application or session layer protocol) are placed in memory buffers and transmitted by TCP in transport Protocol Data Units (for TCP these are usually known as "segments"). The reliable, flow-controlled TCP service is much more complex than UDP, which only provides a Best Effort service. To implement the service, TCP uses a number of protocol timers that ensure reliable and synchronized communication between the two End Systems.

TCP provides reliability with a mechanism called *Positive Acknowledgment with Re-transmission* (PAR). Simply stated, a system using PAR sends the data again, unless it hears from the remote system that the data arrived okay. The unit of data exchanged between cooperating TCP modules is called a segment. Each segment contains a checksum that the recipient uses to verify that the data is undamaged. If the data segment is received undamaged, the receiver sends a positive acknowledgment back to the sender. If the data segment is damaged, the receiver discards it. After an appropriate time-out period, the sending TCP module re-transmits any segment for which no positive acknowledgment has been received.

14.4.1 Aspects of TCP

The main aspects of TCP are:

- **Data transfer**—Data is transmitted between two applications by packaging the data within TCP segments. This data is buffered and forwarded whenever necessary. A push function can be used when the data is required to be sent immediately.
- **Reliability**—TCP uses sequence numbers and positive acknowledgements (ACK) to keep track of transmitted segments. Thus, it can recover from data that is damaged, lost, duplicated, or delivered out of order, such as:
  - □ *Time-outs*—The transmitter waits for a given time (the timeout interval), and if it does not receive an ACK, the data is retransmitted.
  - □ *Sequence numbers*—The sequence numbers are used at the receiver to correctly order the packets and to delete duplicates.
  - □ *Error detection and recovery*—Each packet has a checksum, which is checked by the receiver. If it is incorrect the receiver discards it, and can use the acknowledgements to indicate the retransmission of the packets.
- **Flow control**—TCP returns a window with every ACK. This window indicates a range of acceptable sequence numbers beyond the last segment successfully received. It also indicates the number of bytes that the sender can transmit before receiving further acknowledgements.
- **Multiplexing**—To support multiple connections to a single host, TCP provides a set of ports within each host. This, along with the IP addresses of the source and destination, makes a socket, and a pair of sockets uniquely identifies each connection. Ports are normally associated with various services and allow server programs to listen for defined port numbers.
- **Connections**—A connection is defined by the sockets, sequence numbers and window sizes. Each host must maintain this information for the length of the connection. When the connection is closed, all associated resources are freed. As TCP connections can be made with unreliable hosts and over unreliable communication channels, TCP uses a handshake mechanism with clock-based sequence numbers to avoid inaccurate connection initialization.
- **Precedence and security**—TCP allows for different security and precedence levels. TCP information contains simple acknowledgement messages and a set of sequential numbers. It also supports multiple simultaneous connections using destination and source port numbers, and manages them for both transmission and reception. As with IP, it supports data fragmentation and reassembly and data multiplexing/ demultiplexing.

The set-up and operation of TCP is as follows:

- When a host wishes to make a connection, TCP sends out a request message to the destination machine that contains unique numbers called a socket number, and a port number. The port number has a value that is associated with the application (for example a TELNET connection has the port number 23 and an FTP connection has the port number 21). The message is then passed to the IP layer, which assembles a datagram for transmission to the destination.

- When the destination host receives the connection request, it returns a message containing its own unique socket number and a port number. The socket number and port number thus identify the virtual connection between the two hosts.
- After the connection has been made, the data can flow between the two hosts (called a Data stream).

After TCP receives the stream of data, it assembles the data into packets, called TCP segments. After the segment has been constructed, TCP adds a header (called the protocol data unit) to the front of the segment. This header contains information such as a checksum, the port number, the destination and source socket numbers, the socket number of both machines and segment sequence numbers. The TCP layer then sends the packaged segment down to the IP layer, which encapsulates it and sends it over the network as a datagram.

Ports and Sockets

As previously mentioned, TCP adds a port number and socket number for each host. The port number identifies the required service, whereas the socket number is a unique number for that connection. Thus, a node can have several TELNET connections with the same port number but each connection will have a different socket number. A port number can be any value but there is a standard convention that most systems adopt.

14.4.2 TCP Header Format

The sender's TCP layer communicates with the receiver's TCP layer using the TCP protocol data unit. It defines parameters such as the source port, destination port, and so on as illustrated in figure 14.8.

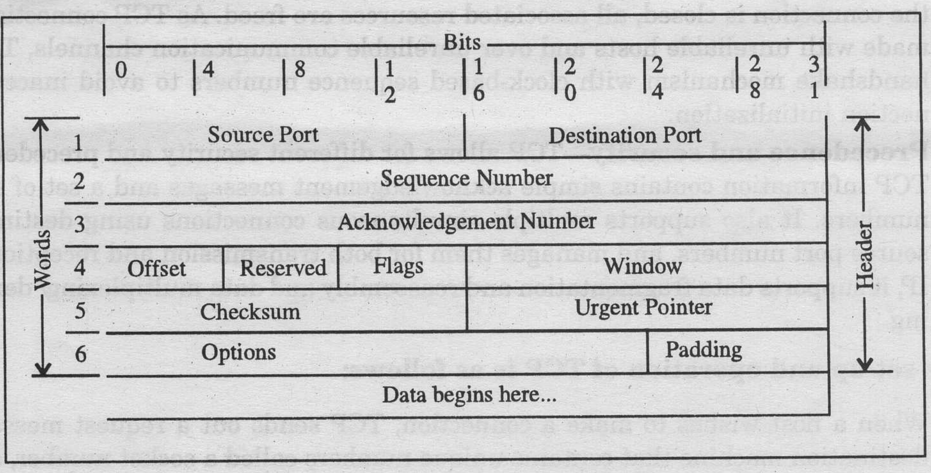


Fig. 14.8 TCP header format

The fields in TCP header format are explained below:

- **Source and destination port number**—These are 16-bit values that identify the local port number (source number and destination port number).
- **Sequence number**—This identifies the current sequence number of the data segment. This allows the receiver to keep track of the data segments received. Any segments that are missing can be easily identified. The sequence number is the first data byte of the DATA segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN + 1.
- **Acknowledgement number**—When the ACK bit is set, it contains the value of the next sequence number the sender of the packet is expecting to receive. This is always set after the connection is made.
- **Data offset**—This is a 32-bit value that identifies the start of the data. It is defined as the number of 32-bit words in the header (as the TCP header always has a multiple number of 32-bit words).
- **Flags**—The flag field is defined as UAPRSF, where U is the urgent flag (URG), A the acknowledgement flag (ACK), P the push function (PSH), R the reset flag (RST), S the sequence synchronize flag (SYN) and F the end-of-transmission flag (FIN). These fields are defined below:
  1. URG (1-bit)—The urgent pointer is valid.
  2. ACK (1-bit)—Makes the acknowledgement number valid.
  3. PSH (1-bit)—High priority data for the application.
  4. RST (1-bit)—Reset the connection.
  5. SYN (1-bit)—Turned on when a connection is being established and the sequence number field will contain the initial sequence number chosen by this host for this connection.
  6. FIN (1-bit)—The sender is done sending data.
- **Window**—This is a 16-bit value and gives the number of data bytes that the receiving host can accept at a time, beginning with the one indicated in the acknowledgement field of this segment.
- **Checksum**—This is a 16-bit checksum for the data and header. It is the 1's complement of the 1's complement sum of all the 16-bit words in the TCP header and data. The checksum is assumed to be a zero when calculating the checksum.
- **UrgPtr**—This is the urgent pointer and is used to identify an important area of data (most systems do not support this facility). It is only used when the URG bit is set. This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment.
- **Padding (variable)**—The TCP header padding is used to ensure that the TCP header ends and data begins on a 32-bit boundary. The padding is composed of zeros.
- **Options**—This is possibly added to segment header.

In TCP, a packet is termed as the complete TCP unit; that is, the header and the data. A segment is a logical unit of data, which is transferred between two TCP hosts. Thus a packet is made up of a header and a segment.

14.4.3 Protocol Operation in Detail

TCP connections contain three phases:

- Connection establishment.
- Data transfer.
- Connection termination.

A 3-way handshake is used to establish a connection. A four-way handshake is used to tear-down a connection. During connection establishment, parameters such as sequence numbers are initialized to help ensure ordered delivery and robustness.

Connection Establishment (3-Way Handshake)

While it is possible for a pair of end hosts to initiate a connection between them simultaneously, typically one end opens a socket and listens passively for a connection from the other. This is commonly referred to as a passive open, and it designates the server-side of a connection. The client-side of a connection initiates an active open by sending an initial SYN segment to the server as part of the 3-way handshake. The server -side should respond to a valid SYN request with a SYN/ACK. Finally, the client-side should respond to the server with an ACK, completing the 3-way handshake and connection establishment phase.

The type of handshake used by TCP is called a three-way handshake because three segments are exchanged. Figure 14.9 shows the simplest form of the three-way handshake. Host *A* begins the connection by sending host *B* a segment with the "Synchronize sequence numbers" (SYN) bit set. This segment tells host *B* that *A* wishes to set up a connection, and it tells *B* what sequence number host *A* will use as a starting number for its segments. (Sequence numbers are used to keep data in the proper order.) Host *B* responds to *A* with a segment that has the "Acknowledgment" (ACK) and SYN bits set. *B*'s segment acknowledges the receipt of *A*'s segment, and informs *A* which Sequence Number host *B* will start with. Finally, host *A* sends a segment that acknowledges receipt of B's segment, and transfers the first actual data.

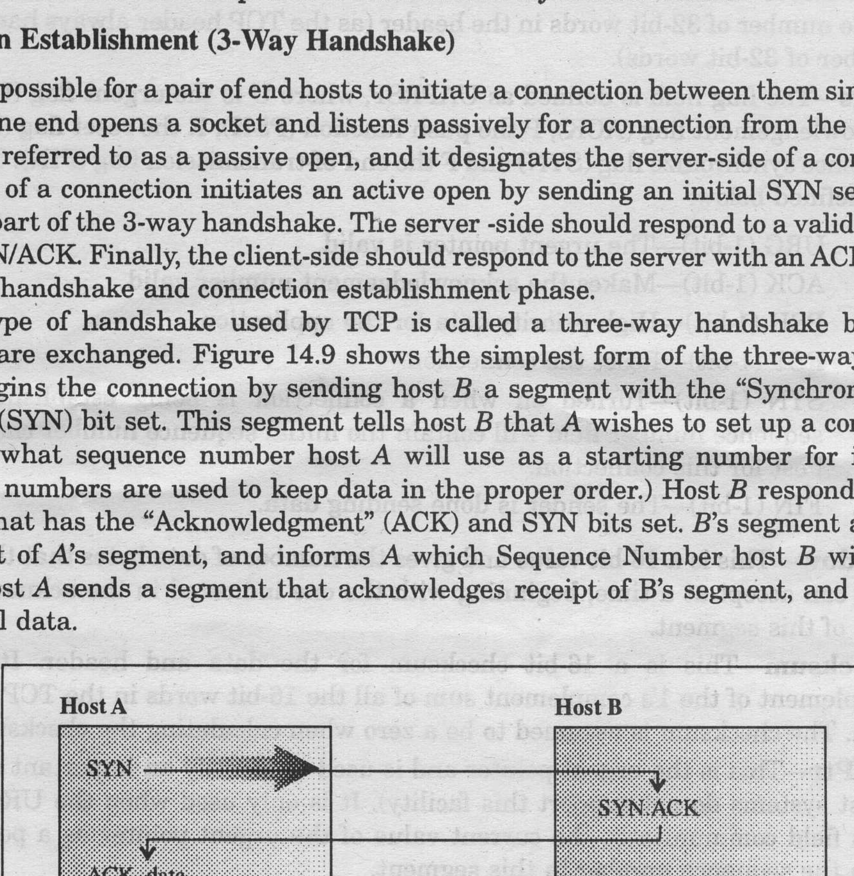


Fig. 14.9 Three-way handshake

Data Transfer

During the data transfer phase, a number of key mechanisms determine TCP's reliability and robustness. These include using sequence numbers for ordering received TCP segments and detecting duplicate data, checksums for segment error detection, and acknowledgements and timers for detecting and adjusting to loss or delay.

During the TCP connection establishment phase, initial sequence numbers (ISNs) are exchanged between the two TCP speakers. These sequence numbers are used to identify data in the byte stream, and are numbers that identify (and count) application data bytes. There is always pair of sequence numbers included in every TCP segment, which is referred to as the sequence number and the acknowledgement number. A TCP sender refers to its own sequence number simply as the sequence number, while the TCP sender refers to receiver's sequence number as the acknowledgement number. To maintain reliability, a receiver acknowledges TCP segment data by indicating it has received up to some location of contiguous bytes in the stream. An enhancement to TCP, called selective acknowledgement (SACK), allows a TCP receiver to acknowledge out of order blocks.

Through the use of sequence and acknowledgement numbers, TCP can properly deliver received segments in the correct byte stream order to a receiving application. Sequence numbers are 32-bit, unsigned numbers, which wrap to zero on the next byte in the stream after 2^{32}-1. One key to maintaining robustness and security for TCP connections is in the selection of the ISN.

A 16-bit checksum, consisting of the one's complement of the one's complement sum of the contents of the TCP segment header and data, is computed by a sender, and included in a segment transmission. (The one's complement sum is used because the end-around carry of that method means that it can be computed in *any* multiple of that length—16-bit, 32-bit, 64-bit, etc. and the result, once *folded*, will be the same.) The TCP receiver recomputes the checksum on the received TCP header and data. The complement was used (above) so that the receiver does not have to zero the checksum field, after saving the checksum value elsewhere; instead, the receiver simply computes the one's complement sum with the checksum in setup, and the result should be –0. If so, the segment is assumed to have arrived intact and without error.

Note that the TCP checksum also covers a 96-bit pseudo header containing the Source Address, the Destınation Address, the Protocol, and TCP length. This provides protection against misrouted segments.

The TCP checksum is a quite weak check by modern standards. Data Link Layers with a high probability of bit error rates may require additional link error correction/detection capabilities. If TCP were to be redesigned today, it would most probably have a 32-bit cyclic redundancy check specified as an error check instead of the current checksum. The weak checksum is partially compensated for by the common use of a CRC or better integrity check at layer 2, below both TCP and IP, such as is used in PPP or the Ethernet frame. However, this does not mean that the 16-bit TCP checksum is redundant: remarkably, surveys of Internet traffic have shown that software and hardware errors that introduce errors in packets between CRC-protected hops are common, and that the end-to-end 16-bit TCP checksum catches most of these simple errors. This is the end-to-end principle at work.

Acknowledgements for data sent, or lack of acknowledgements, are used by senders to implicitly interpret network conditions between the TCP sender and receiver. Coupled with timers, TCP senders and receivers can alter the behavior of the flow of data. This is more generally referred to as flow control, congestion control and/or congestion avoidance. TCP uses a number of mechanisms to achieve high performance and avoid congesting the network (i.e., send data faster than either the network or the host on the other end, can utilize it). These mechanisms include the use of a sliding window, the slow-start algorithm, the congestion avoidance algorithm, the fast retransmit and fast recovery algorithms and more. Enhancing TCP to reliably handle loss, minimize errors, manage congestion and go fast in very high-speed environments are ongoing areas of research and standards development.

TCP window size

The TCP receive window size is the amount of received data (in bytes) that can be buffered during a connection. The sending host can send only that amount of data before it must wait for an acknowledgment and window update from the receiving host. The Windows TCP/IP stack is designed to self-tune itself in most environments and uses larger default window sizes than earlier versions.

Windows scaling

For more efficient use of high bandwidth networks, a larger TCP window size may be used. The TCP window size field controls the flow of data and is limited to 2-bytes or a window size of 65,535 bytes. Since the size field cannot be expanded, a scaling factor is used. TCP window scale is an option used to increase the maximum window size from 65,535 bytes to 1 Gigabyte. The window scale option is used only during the TCP 3-way handshake. The window scale value represents the number of bits to left-shift the 16-bit window size field. The window scale value can be set from 0 (no shift) to 14.

Connection termination

The connection termination phase uses a four-way handshake, with each side of the connection terminating independently. Therefore, a typical teardown requires a pair of FIN and ACK segments from each TCP endpoint.

Alternatives to TCP

However, TCP is not appropriate for many applications, and newer transport layer protocols are being designed and deployed to address some of the inherent weaknesses. For example, real-time applications often do not need and will suffer from TCP's reliable delivery mechanisms. In those types of applications it is often better to deal with some loss, errors or congestion than to try to adjust for them. Example applications that do not typically use TCP include real-time streaming multimedia (such as Internet radio), real-time multiplayer games and voice over IP (VoIP). Any application that does not require reliability or that wants to minimize functionality, may choose to avoid using TCP. In many cases, the User Datagram Protocol (UDP) may be used in place of TCP when just application multiplexing services are required.

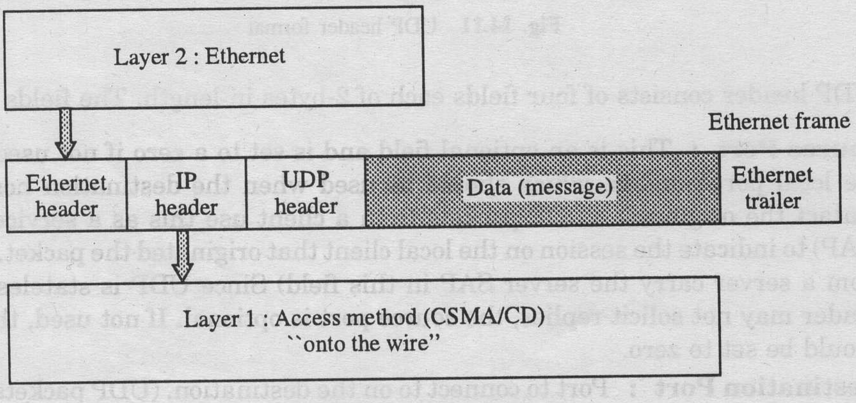
14.5 User Datagram Protocol (UDP)

TCP allows for a reliable connection-based transfer of data. The User Datagram Protocol (UDP) is an unreliable connection-less approach, where datagrams are sent into the network without any acknowledgements or connections (and thus relies on high-level protocols to provide for these). The User Datagram Protocol gives application programs direct access to a datagram delivery service, like the delivery service that IP provides. This allows applications to exchange messages over the network with a minimum of protocol overhead.

The **User Datagram Protocol (UDP)** is one of the core protocols of the Internet protocol suite. Using UDP, programs on networked computers can send short messages known as datagrams to one another. UDP does not provide the reliability and ordering guarantees that TCP does; datagrams may arrive out of order or go missing without notice. However, as a result, UDP is faster and more efficient for many lightweight or time-sensitive purposes. Common network applications that use UDP includes the Domain Name System (DNS), streaming media applications, Voice over IP, and online games. UDP is widely used for streaming audio and video, voice over IP (VoIP) and videoconferencing, because there is no time to retransmit erroneous or dropped packets.

The service provided by UDP is an unreliable service that provides no guarantees for delivery and no protection from duplication (e.g., if this arises due to software errors within an Intermediate System (IS)). "Unreliable" merely means that there are no techniques in the protocol for verifying that the data reached the other end of the network correctly. Within your computer, UDP will deliver data correctly. The simplicity of UDP reduces the overhead from using the protocol and the services may be adequate in many cases.

It is a protocol within the TCP/IP protocol suite that is used in place of TCP when a reliable delivery is not required. There is less processing of UDP packets than there is for TCP. If UDP is used and a reliable delivery is required, packet sequence checking and error notification must be written into the applications. UDP is "connectionless" and does not use a handshake to start a session. It just sends out packets.



UDP within an ethernet frame

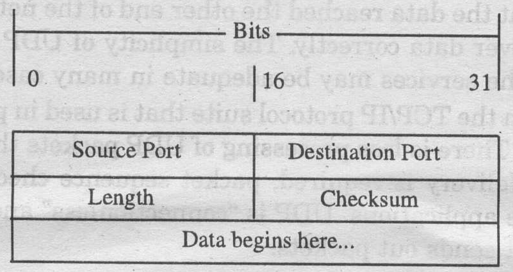
Fig. 14.10 A UDP packet is framed just like a TCP packet.
This shows a UDP packet in an Ethernet frame ready for transmission over the network.

Lacking reliability, UDP applications must generally be willing to accept some loss, errors or duplication. Some applications such as TFTP may add rudimentary reliability mechanisms into the application layer as needed. Most often, UDP applications do not require reliability mechanisms and may even be hindered by them. If an application requires a high degree of reliability, a protocol such as the Transmission Control Protocol may be used instead.

Lacking any congestion avoidance and control mechanisms, network-based mechanisms are required to minimize potential congestion collapse effects of uncontrolled, high rate UDP traffic loads. In other words, since UDP senders cannot detect congestion, network-based elements such as routers using packet queueing and dropping techniques will often be the only tool available to slow down excessive UDP traffic. The Datagram Congestion Control Protocol (DCCP) is being designed as a partial solution to this potential problem by adding end host congestion control behavior to high-rate UDP streams such as streaming media.

14.5.1 UDP Header Format

A computer may send UDP packets without first establishing a connection to the recipient. The computer completes the appropriate fields in the UDP header (PCI) and forwards the data together with the header for transmission by the IP network layer. The UDP header consists of only 4 header fields of which two are optional. The source and destination port fields are 16-bit fields that identify the sending and receiving process.



Source Port	Destination Port
Length	Checksum
Data begins here...	

Fig. 14.11 UDP header format

The UDP header consists of four fields each of 2-bytes in length. The fields are:

- **Source Port :** This is an optional field and is set to a zero if not used. It identifies the local port number, which should be used when the destination host requires to contact the originator. (UDP packets from a client use this as a service access point (SAP) to indicate the session on the local client that originated the packet. UDP packets from a server carry the server SAP in this field) Since UDP is stateless and a UDP sender may not solicit replies, the source port is optional. If not used, the source port should be set to zero.
- **Destination Port :** Port to connect to on the destination. (UDP packets from a client use this as a service access point (SAP) to indicate the service required from the remote server. UDP packets from a server carry the client SAP in this field).
- **UDP length :** Number of bytes in the datagram, including the UDP header and the

data. (The number of bytes comprising the combined UDP header information and payload data) The port fields are followed by a mandatory length field specified as bytes of the UDP datagram including the data. The minimum value of the length field is 8 (octets).

- **UDP Checksum :** The 16-bit 1's complement of the 1's complement sum of the IP header, the UDP header, the data (which, if necessary, is padded with zero bytes at the end, to make an even number of bytes). (A checksum to verify that the end-to-end data has not been corrupted by routers or bridges in the network or by the processing in an end system. The algorithm to compute the checksum is the Standard Internet Checksum algorithm. If this check is not required, the value of 0 × 0000 is placed in this field, in which case the data is not checked by the receiver.) The remaining header field is a 16-bit checksum field covering the header and data. The checksum is also optional, but almost always used in practice.

Like for other transport protocols, the UDP header and data are not processed by Intermediate Systems (IS) in the network, and are delivered to the final destination in the same form as originally transmitted.

At the final destination, the UDP protocol layer receives packets from the IP network layer. These are checked using the checksum (when > 0, this checks correct end-to-end operation of the network service) and all invalid PDUs are discarded. UDP does not make any provision for error reporting if the packets are not delivered. Valid data are passed to the appropriate session layer protocol identified by the source and destination port numbers (i.e., the session service access points).

Generally, clients set the source port number to a unique number that they choose themselves—usually based on the program that started the connection. Since this number is returned by the server in responses, this lets the sender know which "conversation" incoming packets are to be sent to. The destination port of packets sent by the client is usually set to one of a number of well-known ports. These usually correspond to one of a number of different applications, e.g., port 23 is used for telnet, and port 80 is used for web servers.

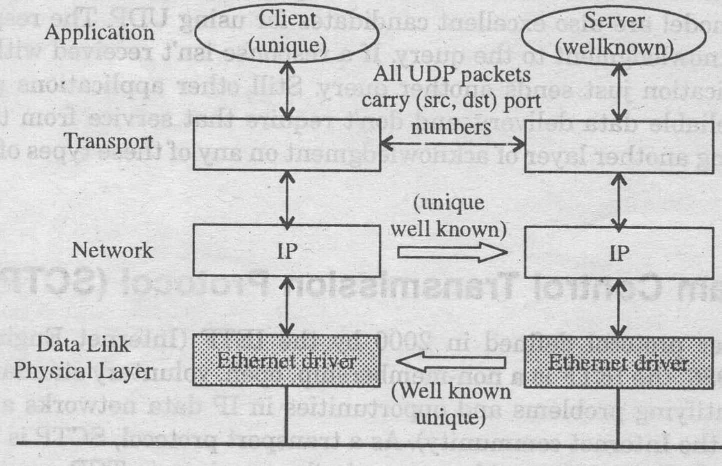


Fig. 14.12 UDP transmitting packets over an IP network using ethernet

A server process (program), listens for UDP packets received with a particular well-known port number and tells its local UDP layer to send packets matching this destination port number to the server program. It determines which client these packets come from by examining the received IP source address and the received unique UDP source port number. Any responses which the server needs to send to back to a client are sent with the source port number of the server (the well-known port number) and the destination port selected by the client.

14.5.2 Applications

UDP is used when hosts do not require making a connection with the other side, and where reliability is built into a high-layer protocol. It is also used when there is no segmentation of data (as there are no segment values). Some applications are solely TCP or solely UDP, whereas the rest can use either. For example:

- **TCP applications :** FTP, TELNET, SMTP, FINGER, DNS and LOGIN.
- **UDP applications :** RIP, TFTP, NFS and SNMP.
- **TCP or UDP applications :** HTTP, POP-3 and ECHO.

Applications of UDP include:

- **DNS :** Supports domain name mapping to IP addresses.
- **NFS :** Supports a distributed file system.
- **SNMP :** Supports network management for network devices.
- **TFTP (Trivial FTP) :** A simplified version of FTP (typically used to update computers/routers with firmware updates).

Applications programmers choose UDP as a data transport service for a number of reasons. If the amount of data being transmitted is small, the overhead of creating connections and ensuring reliable delivery may be greater than the work of re-transmitting the entire data set. In this case, UDP is the most efficient choice for a Transport Layer protocol. Applications that fit a query-response model are also excellent candidates for using UDP. The response can be used as a positive acknowledgment to the query. If a response isn't received within a certain time period, the application just sends another query. Still other applications provide their own techniques for reliable data delivery, and don't require that service from the transport layer protocol. Imposing another layer of acknowledgment on any of these types of applications is inefficient.

14.6 The Stream Control Transmission Protocol (SCTP)

It is a transport layer protocol defined in 2000 by the IETF (Internet Engineering Task Force—Founded in 1986, the IETF is a non-membership, open, voluntary standards organization dedicated to identifying problems and opportunities in IP data networks and proposing technical solutions to the Internet community). As a transport protocol, SCTP is equivalent in a sense to TCP or UDP. Indeed it provides some similar services as TCP, ensuring reliable, in-sequence transport of messages with congestion control. While TCP is byte-oriented, SCTP deals with framed messages.

SCTP is a reliable transport protocol operating on top of a potentially unreliable connectionless packet service such as IP. It offers acknowledged error-free non-duplicated transfer of datagrams (messages). Detection of data corruption, loss of data and duplication of data is achieved by using checksums and sequence numbers. A selective retransmission mechanism is applied to correct loss or corruption of data.

SCTP can be used as the transport protocol for applications where monitoring and detection of loss of session is required. For such applications, the SCTP path/session failure detection mechanisms, especially the heartbeat, will actively monitor the connectivity of the session.

A major contribution of SCTP is multi-homing support, where one (or both) endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between hosts or network cards.

It offers the following services to its users:

- Acknowledged error-free non-duplicated transfer of user data.
- Data fragmentation to conform to discovered path MTU size.
- Sequenced delivery of user messages within multiple streams, with an option for order-of-arrival delivery of individual user messages.
- Optional bundling of multiple user messages into a single SCTP packet.
- Network-level fault tolerance through supporting of multi-homing at either or both ends of an association.

The design of SCTP includes appropriate congestion avoidance behavior and resistance to flooding and masquerade attacks.

14.7 The Datagram Congestion Control Protocol (DCCP)

It is a message-oriented transport layer protocol that is currently under development in the IETF. Applications that might make use of DCCP include those with timing constraints on the delivery of data such that reliable in-order delivery, when combined with congestion control, is likely to result in some information arriving at the receiver after it is no longer of use. Such applications might include streaming media and Internet telephony. Congestion control is the way that a network protocol discovers the available network capacity on a particular path. The primary motivation for the development of DCCP is to provide a way for such applications to gain access to standard congestion control mechanisms without having to implement them at the application layer.

DCCP is intended for applications that require the flow-based semantics of TCP, but which do not want TCP's in-order delivery and reliability semantics, or which would like different congestion control dynamics than TCP. Similarly, DCCP is intended for applications that do not require features of SCTP such as sequenced delivery within multiple streams.

To date most such applications have used either TCP, with the problems described above, or used UDP and implemented their own congestion control mechanisms (or no congestion control at all). The purpose of DCCP is to provide a standard way to implement congestion control and congestion control negotiation for such applications. One of the motivations for DCCP is to enable the use of ECN (Explicit Congestion Notification—explained below), along with conformant end-to-end congestion control, for applications that would otherwise be using UDP. In addition, DCCP implements reliable connection setup, teardown, and feature negotiation.

A DCCP connection contains acknowledgement traffic as well as data traffic. Acknowledgements inform a sender whether its packets arrived, and whether they were ECN marked. Acks are transmitted as reliably as the congestion control mechanism in use requires, possibly completely reliably.

IP ECN (Explicit Congestion Notification)

ECN is only used when the two hosts signal that they want to use it. With this method, an ECN bit is used to signal that there is explicit congestion. This is better in some ways than the indirect packet delete congestion notification but it requires explicit support by both hosts to be effective. Some outdated or buggy network equipment drops packets with the ECN bit set, rather than ignoring the bit.

● ● ●

15

Session And Presentation Layer

Introduction to the Session Layer

The fifth layer in the OSI Reference Model is the session layer. It is the lowest of the three upper layers, which collectively are concerned mainly with software application issues and not with the details of network and Internet implementation.

The name of this layer tells you much about what it is designed to do : to allow devices to establish and manage sessions. In general terms, a session is a persistent logical linking of two software application processes, to allow them to exchange data over a prolonged period of time. In some discussions, these sessions are called dialogs; they are roughly analogous to a telephone call made between two people.

The term "session" is somewhat vague, and this means that there is sometimes disagreement on the specific functions that belong at the session layer or even whether certain protocols belong at the session layer or not. To understand the meaning of session let us first differentiate between a "connection" and a "session". Connections are normally the province of layer four and layer three, yet a Transmission Control Protocol (TCP) connection, for example, can persist for a long time. The longevity of TCP connections makes them hard to distinguish from "sessions".

So we can say that a connection between two presentation layer processes is called a session. A presentation entity (PE) can only communicate with another PE through initiating or accepting a session connection. There may be a number of concurrent or consecutive session connections between the two PEs.

A session is a series of related connection-oriented transmissions between network nodes.

Another way to look at it is that a session is the interrelated communications between two or more presentation entities, which emphasizes that the Session layer provides services to the Presentation layer.

The session layer resides above the transport layer, and provides "value added" services to the underlying transport layer services. The session layer (along with the presentation layer) add services to the transport layer that are likely to be of use to applications, so that each application doesn't have to provide its own implementation.

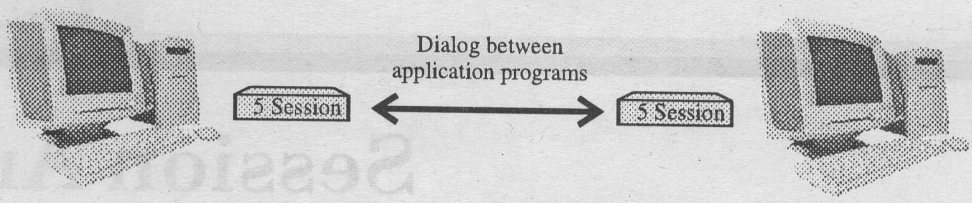


Fig. 15.1 Layer 5-Session Layer

Administration of Session in Session Layer

A session is a formal dialog between a service requester and a service provider. Sessions have at least three phases:

- **Connection establishment** : In this phase, a service requester requests initiation of a service. During the setup process, communication is established and rules are agreed upon. The connection establishment phase establishes the parameters for the communication session. Actually, the connection establishment phase is comprised of several tasks, including the following:
    - ¤ Specification of required services.
    - ¤ User login authentication and other security procedures.
    - ¤ Negotiation of protocols and protocol parameters.
    - ¤ Notification of connection Ids.
    - ¤ Establishment of dialog control, as well as acknowledgment of numbering and retransmission procedures.
- **Data transfer** : After the connection is established, the devices involved can initiate a dialog (data transfer phase). Besides exchanging data, these devices exchange acknowledgments and other control data that manage the dialog. Because of the rules agreed upon during setup, each party to the dialog knows what to expect. Communication is therefore efficient, and errors are easy to detect. The Session layer also can incorporate protocols to resume dialogs that have been interrupted. After a formal dialog has been established, devices recognize a lost connection whenever the connection has not been formally released. Therefore, a device realizes that a connection has been lost when the device fails to receive an expected acknowledgment or data transmission.

    Data transfer is provided in four forms : normal, expedited, capability and typed.

Normal data transfer is the ordinary data delivery method. **Expedited data** has some priority and is able to bypass some restrictions that might impede the transfer of normal data. Expedited data is restricted to small packets, whereas normal data has no such restriction. **Typed** and **capability data** types are specific to the Session Layer. They are used principally to permit overrides of dialog control mechanisms that may be in effect.

- **Connection release :** When the session is completed, the dialog is terminated in an orderly fashion. Within a certain time period, two devices can reenter the session that was interrupted but not released. The connection release phase is an orderly process that shuts down communication and releases resources on the service provider.

In connectionless mode the only purpose of the session layer is to provide a one-to-one mapping of transport to session addresses.

15.1 Design Issues Of Session Layer

The session layer provides the following services:

1. Dialog Management

A dialog is defined to be the series of sessions used for a complex process or transfer of a large quantity of data. Dialog management is basically deciding whose turn it is to talk. Some applications operate in half-duplex mode, whereby the two sides alternate between sending and receiving messages and never send data simultaneously.

In the ISO protocols, dialog management is implemented through the use of a data token. The token is sent back and forth and a user may transmit only when it possesses the token. The Session layer manages dialogs between two computers by establishing, managing, and terminating communications. As illustrated in Figure 15.2, dialogs can take three forms:

- **Simplex dialogs :** These dialogs are responsible for only one-way data transfers. An example is a fire alarm, which sends an alarm message to the fire station but cannot (and does not need to) receive messages from the fire station.
- **Half-duplex dialogs :** These dialogs handle two-way data transfers in which the data flows in only one direction at a time. When one device completes a transmission, this device must "turn over" the medium to the other device so that this second device has a turn to transmit. For example, CB radio operators, converse on the same communication channel. When one operator is finished transmitting, he must release his transmit key so that the other operator can send a response.
- **Full-duplex dialogs :** This third type of dialog permits two-way simultaneous data transfers by providing each device with a separate communication channel. Voice telephones are full-duplex devices, and either party to a conversation can talk at any time. Most computer modems can operate in full-duplex mode.

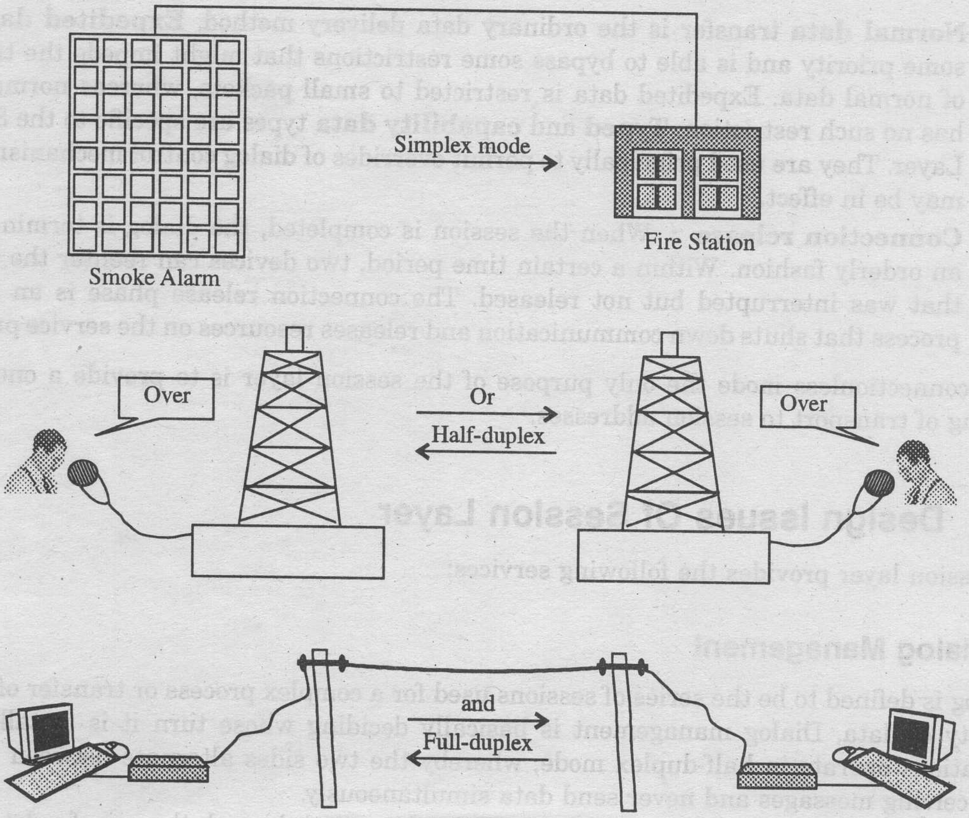


Fig. 15.2 Simplex and duplex communication modes

The Session layer also marks the data stream with checkpoints and monitors the receipt of those checkpoints. In the event of a failure, the sending PC can retransmit starting with the data sent after the last checkpoint, rather than resending the whole message.

2. Synchronization

It is basically moving the two session entities into a known state. The transport layer handles only communication errors, synchronization deals with upper layer errors. In a file transfer, for instance, the transport layer might deliver data correctly, but the application layer might be unable to write the file because the file system is full.

Users can split the data stream into pages, inserting synchronization points between each page. When an error occurs, the receiver can resynchronize the state of the session to a previous synchronization point. This requires that the sender hold data as long as may be needed.

Synchronization is achieved through the use of sequence numbers. The ISO protocols provide both major and minor synchronization points. When resynchronizing, one can only go back as far as the previous major synchronization point. In addition, major synchronization points are acknowledged through explicit messages (making their use expensive). In contrast, minor synchronization points are just markers. So following synchronization methods can be defined:

- **Initial synchronization** : Occurs when the session ID and start sequence numbers are generated.
- **Major synchronization** : During reset of session, it reestablishes ID and sequence numbers.
- **Minor synchronization** : Fine tunes in existing sessions, assumes prior data is okay.
- **Resynchronization**
  - □ Restart but use checkpoints in prior data.
  - □ Abandon—drop session and reinitiate, discarding data.

3. Activity Management

It allows the user to delimit data into logical units called activities. Each activity is independent of activities that come before and after it, and an activity can be processed on its own. Activities might be used to delimit files of a multi-file transfer.

Activities are also used for quarantining, collecting all the messages of a multi-message exchange together before processing them. The receiving application would begin processing messages only after all the messages had arrived. This provides a way of helping to insure that all or none of a set of operations are performed.

For example, a bank transaction may consist of locking a record, updating a value, and then unlocking the record. If an application processed the first operation, but never received the remaining operations (due to client or network failures), the record would remain locked forever. Quarantining addresses this problem.

Quarantine service allows for a sending PE (Presentation Entity) to request that an integral number of SSDUs should not be made available to the receiving PE until explicitly released by the sending PE. This is transparent to the receiving PE and some or all data may be discarded by the sending PE without it knowing.

4. Exception Handling

A general purpose mechanism for reporting errors. It allows for PEs to be notified of unrecoverable session malfunctions and other problems not covered by other services.

The collection of services available on a particular connection is subject to selection by the Presentation entity that establishes the connection. Specific Session-Layer services are grouped together into functional units. Functional units define a number of frequently chosen operational styles and their selection is a matter of negotiation at the time of connection establishment.

World-Wide Web applications often require large files to travel across the network. Session Layer services could be employed to assist in recovery of interrupted transfers. The synchronization issue is dealt with by the client/server mode of operation. The client requests a file from the server and the server sends it if possible. The client then goes on with other processing. Synchronization between the client and server is not an issue in this model.

15.2 Session Layer Protocols

The session layer implementation of the OSI protocol suite consists of a session protocol and a session service. The session protocol allows session-service users (SS-users) to communicate with the session service. An SS-user is an entity that requests the services of the session layer. Such requests are made at session-service access points (SSAPs), and SS-users are uniquely identified by using an SSAP address.

Session service provides four basic services to SS-users.

1. It establishes and terminates connections between SS-users and synchronizes the data exchange between them.
2. It performs various negotiations for the use of session layer tokens, which the SS-user must possess to begin communicating.
3. It inserts synchronization points in transmitted data that allow the session to be recovered in the event of errors or interruptions.
4. Finally, it enables SS-users to interrupt a session and resume it later at a specific point.

The following services and protocols are defined on the Sessions layer:

- RPC (Remote Procedure Call).
- ASP (AppleTalk Session Protocol).
- SQL (Structured Query Language).
- NFS (Network File Services).
- SCP (Serial Communications Protocol).
- ZIP (AppleTalk Zone Information Protocol).

From the above list of protocols we will discuss RPC in detail.

15.2.1 Remote Procedure Call (RPC)

RPC is a powerful technique for constructing distributed, client-server based applications. It is based on extending the notion of conventional, or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports.

RPC makes the client/server model of computing more powerful and easier to program. RPC was developed by Sun Microsystems, and is a collection of tools and library functions.